DAIMLERCHRYSLER

KEYWORD PROTOCOL 2000

REQUIREMENTS DEFINITION

RELEASE 2.2

05 AUGUST 2002

VEHICLE DIAGNOSTICS, COC
ELECTRICAL DISTRIBUTION SYSTEMS DEPARTMENT
DAIMLERCHRYSLER-AUBURN HILLS (DCA)
AUBURN HILLS, MI USA

DIAGNOSTIC STANDARDS
GSP / SDE
DAIMLERCHRYSLER-STUTTGART (DCS)
SINDELFINGEN, GERMANY

ELECTRONICS DESIGN
ELECTRICAL / ELECTRONICS SYSTEMS DEPARTMENT
MITSUBISHI MOTORS CORPORATION (MMC)
OKAZAKI, JAPAN

CONTACT LIST

	Contact Name	Department	Location	Phone Number	Email Address
Primary	Edmond T. Delude	Vehicle Diagnostics, CoC	DCA	(+1) 248-576-3178	ETD1@DaimlerChrysler.com
	Gunnar Gaisser	Diagnostic Standards	DCS	(+49) 7031-90-70327	gunnar.gaisser@DaimlerChrysler.com
	Toshiki Fukaya	Electronics Design	MMC	(+81) 564-325228	fukaya-toshiki@pde.mitsubishi-motors.co.jp
Secondary	Werner Preuschoff	Diagnostic Standards	DCS	(+49) 7031-90-70337	werner.preuschoff@DaimlerChrysler.com
	Michael M. Flaherty	Vehicle Diagnostics, CoC	DCA	(+1) 248-576-0607	MMF6@ DaimlerChrysler.com
	Thomas R. Tasky	Vehicle Diagnostics, CoC	DCA	(+1) 248-576-3245	TRT1@ DaimlerChrysler.com
Other					

DC-GROUP INFORMATION

VEHICLE DIAGNOSTICS, CoC
ELECTRICAL DISTRIBUTION SYSTEMS DEPARTMENT
DAIMLERCHRYSLER-AUBURN HILLS (DCA)

CIMS 484-02-15
800 CHRYSLER DRIVE EAST
AUBURN HILLS, MICHIGAN 48326-2757 USA

DAIMLERCHRYSLER INTRANET
http://corediag.scp.chrysler.com

DIAGNOSTIC STANDARDS
GSP / SDE
DAIMLERCHRYSLER-STUTTGART (DCS)

HPC X942
D-71059 SINDELFINGEN, GERMANY

DAIMLERCHRYSLER INTRANET
http://INTRA-DIAGNOSTICS.DAIMLERCHRYSLER.COM/

CHANGE LOG

Date	Release	Description	Author
02-Feb-01	2.0	Official Release Note for DCA: For a list of modifications between version 1.0 and 2.0, please refer http://corediag.scp.chrysler.com.; KWP2K_1.0_TO_2.0_CHANGELOG Note for DCS: For a list of modifications between version 1.1 and 2.0, please refer to "Compatibility Hints (2.0 vs. 1.1)" on the EP/EID Diagnostic Standards homepage	ETD / GG
20 July 2001	2.1	**Major Changes:** Added diagnostic service Control DTC Setting ($85) [All] Added $18 $00 and $18 $01 [DCA] Added $1A $89 [MMC] Removed NonVolatile Memory Reset ($11 $82) [All] Added Vehicle Configuration Information ($21 $E5) [DCA] Added Flash Info 1 / 2 ($21 $E6 / $E7) [DCX] Added System Diagnostic General Parameter Data ($21 $E8) [DCS] Added System Diagnostic Global Parameter Data ($21 $E9) [DCS] Added Tell-Tale Retention Stack Routine ($31 $E2), ($32 $E2), ($33 $E2) [DCA] Added routine local Ids $E3, $E4 to request DTCs and Environment Data from a protected error memory [DCS] Added routine local Ids $E5, $E6 to request Event Numbers and corresponding Environment Data from a protected error memory [DCS] Added additional services in Stand By Session [DCS] DBCom Data ($21 $E2) extended for reprogramming [DCS] Clarified Message Usage requirements that indicate Conditional with the condtion the would cause the byte to become mandatory. [All] Changed Group of DTC parameter option All DTC's ($FF00) to Mandatory for services $14 and $18 [All] Modified $1A $87 to extend Software Version from 2 to 3 bytes [All] For Tester Present ($3E), changed Response Required = No Response Required ($02) to Mandatory [All] **Minor Changes:** Title page changed according to Corporate Design Added Mitsubishi as additional user of specification Updated Diagnostic Teams section to include MMC Updated Figure 1 to include MMC Added additional information for clarification in No Response section. Updated functional groups in the Diagnostic Service Identifiers to include new services Renamed DCX-Diagnostic-Session($10 $92) to Extended Diagnostic Session($10 $92) Renamed present/stored to active/stored Figure 5 (NRC) updated Description of NRC$23 for legacy purposes added (with note not to use) Corrected Implementation Example of $1A $87 For $31 explanation how to handle "long running" routines Note to use variantcoding with $21, $3B added ECU programming session renamed to ECU-Reprogramming Session Updated all request and response messages to begin with byte 0. Updated all LID's description's so byte # matches byte position in request / resp. message. Added System Supplier Specific Range to Supported Diagnositc Services in Table 3.1.1-1 Modified reference section. Added example of implementation for $18 $00 Changed Example for $18 02, $18 03	ETD / GG

Date	Release	Description
5 Aug 02	2.2	**Major Changes:** Added Diagnostic Service $2E (3.15) Added Diagnostic Service $22 (3.9) Added NRC $A0 (5.23) Added NRC $A1 (5.24) Added Define by Memory Address ($02) for Dynamically Defined Local Identifier (3.14.2/4.4.5) Added Define by Identifier ($03) for Dynamically Defined Local Identifier (3.14.2/4.4.5) Added Requirement #3 (3.16.1) Added $31 $E7 (3.17.1/4.10.5) Added $31 $E8 (3.17.1/4.10.5) Added $31 $E9 (3.17.1/4.10.5) Added Condition #3 (3.17.2) Added Requirement #9 (Note: Behavior already seen in Vector Diagnostic Software Modules.) (3.2.1) Added Read Data By Identifier($22) and Response On Event ($86) to “Normal Default Session” in table 3.2.1-1 (3.2.1) Added Read Data By Local Identifier($21), Read Data By Identifier ($22), Read Memory By Address ($23) to “ECU Flash Reprogramming Session” in table 3.2.1-1 (3.2.1) Changed Stand By Session ($10 $89) and ECU Passive Session ($10 $90) from Optional to Conditional. (3.2.1) Added Condition #2 for Stand By Session ($10 $89) and ECU Passive Session ($10 $90) (3.2.1) Added Condition #1 (3.22.2) Added option for Block Sequence Counter(BSC) in Transfer Data (3.22.2/4.11) Updated Positive Response Format to accommodate BSC (3.22.3) Added Requirement #2 (3.27.1) Modified all Requirements for Response On Event ($86) (3.28.1) Added Event Type $A0 (3.28.2/4.12) Added Non volatile memory reset ($11 $82) (3.3.1/4.2.1) Added Requirement #7 (3.6.1) Added Condition #4 to support $E0 parameter (3.6.2) Added Request Extended Number of Supported DTCs ($E0) (3.6.2/4.3.4) Added Positive Response Format Table (3.6.4-1) for $E0 (3.6.4) Added LID $E4 for ECU Flash Reprogramming (3.8.2/4.4.4) Added LID $EA for ECU Configuration (3.8.2/4.4.4) Added LID $EB for Diagnostic Protocol Information (3.8.2/4.4.4)

Date	Release	Description
5 Aug 02	2.2	**Minor Changes:** Updated “Diagnostic Data Manipulation” list of services to include $22/$2E (3) Added Note regarding DCA Security Specification in Purpose (3.11) Added clarification to the Purpose (3.26) Renamed Section 4.10 (4.1) Added “Request Transfer Exit ($37)” to “APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S)” (5.13) Added “Request Transfer Exit ($37)” to “APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S)” (5.14) Added “Request Transfer Exit ($37)” to “APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S)” (5.15) Updated definition of Conditional (2.1.1) Clarified “No Response” behavior (2.1.4) Updated Table 3.1.1-1 to include $22/$2E (3.1.1) Corrected Seed Length in Example (3.11.9) Updated Requirement #2 (3.16.1) Removed $31 $E2 (3.17.1) Modified Condition #2 (3.17.2) Removed $32 $E0 and $E1 (3.18.1) Removed $32 $E2 (3.18.1/4.10.5) Modified Condition #2 (3.19.2) Added clarification to Purpose (3.2.1) Amended Requirement #2 to accommodate $10 $90 (3.2.1) Added clarification to Requirement #6 (3.2.1) Added clarification to Requirement #8 (3.2.1) Added $22/$2E and “Supplier Specific Range” to table 3.2.1-1 (3.2.1) Fixed typo in table 3.21.4-1 (3.21.4) Re-worded Requirement #1 (3.26.1) Added clarification to Requirement # 4 (3.5.1) Changed Bytes 2,3, and 4 from Mandatory to Conditional (3.5.3) Added Condition #1 to accommodate the change to bytes 2, 3, and 4 (3.5.3) Changed from Condition to Condition #2 and added clarification (3.5.3) Modified Heading Description for 3.6.3 (3.6.3) For Condition, changed from “stored” to “requested” (3.6.3) Added Example for $E0 (3.6.9) Updated example 3.6.9-2 (3.6.9) Added conditions 5 and 6 (3.8.2) Added Clarification to ECU Passive Session (4.1.1) Added Clarification to ECU Standby Session (4.1.1) Added Clarification to Extended Diagnostic Session (4.1.1) Removed “during the last drive cycle” for DTC Readiness Flag (4.3.6) Updated Figure 5 and modified Note 2.(4.3.6) Updated “Diagnostic Information “ (byte 4) (4.4.4) Updated “Diagnostic Information “ (byte 5) (4.4.4) Changed Model Year from BCD to HEX encoding for $21 $E5 (4.4.4) Updated Table 4.4.4-16 (4.4.4) Updated Appendix A (Appendix A) Removed Appendix B and created the “DCX MMC Common State Encoding Tables” (Appendix B) Updated Contact Info (Contact Info) Updated author information (Cover page) Updated DC Group Info (DC Group Info) Changed Operating System to Reserved for LID $E0 (4.4.4) Updated $21 $E8 and $21 $E9 (4.4.4)

TABLE OF CONTENTS

1 INTRODUCTION

1.1 OVERVIEW

1.1.1 KEYWORD PROTOCOL 2000

The Keyword Protocol 2000 (KWP 2000) Design Standard defines the diagnostic communication protocol requirements comprising the diagnostic services, data parameter identifiers, and request/response message format exchange used between off-board diagnostic tools and on-board Electronic Control Units (ECU's). This specification provides recommended uniformity and guidance for incorporating diagnostic protocol functionality, as well as requirements governing consistent methods for implementation.

1.1.2 GOAL OF THIS DOCUMENT

The goal of this document is to specify the requirements for proper implementation of Keyword Protocol 2000. This standard was created to address the need for internal standardization of diagnostic services as applied to all DaimlerChrysler Corporation (DCX) and Mitsubishi Motors Corporation (MMC) vehicle applications. It is intended for use by electrical/electronic development engineers, system engineers, and suppliers in the design, development, and implementation of diagnostic services for ECU's.

1.1.3 REQUIREMENTS

Adherence to this design standard is absolutely necessary to adequately support diagnostic requirements for engineering, manufacturing, and service, and therefore must be strictly enforced. The following requirements must be satisfied for compliance:

- Keyword Protocol 2000 is required to support basic and enhanced diagnostic services for all ECU's.
- Keyword Protocol 2000 is required to replace all legacy protocols (i.e. SCI, CCD, KWFB, and SAE J2190 Enhanced Diagnostic Protocols.)
- All ECU's with diagnostic capability must conform to this standard in its entirety.

Diagnostic services detailed herein are primarily focused on "mandated" diagnostic requirements. However, this specification does allow provisions for "recommended" and "optional" diagnostic services to be implemented at the discretion of the ECU supplier and ECU development engineer.

1.2 DEVELOPMENT HISTORY OF KWP 2000 DESIGN STANDARD

1.2.1 DOCUMENT HISTORY

This document is based on existing diagnostic standards including the International Standard ISO 14230-3 – "Implementation of Diagnostic Services Recommended Practice", and the Daimler-Benz internal corporate standard DB-KWP2000 – "Serielle Diagnose-Schnittstelle fuer Elektronische Steuergeraete". Refer to Figure 1-1 for the hierarchy of documents used in the development of this document. A complete list of reference documents may be found in "Appendix A – References"

1.2.2 DIAGNOSTIC TEAMS

This document has been developed with the intent that it shall be used by all DaimlerChrysler Corporation (DCX) and Mitsubishi Motors Corporation (MMC) vehicle applications. In order to achieve this objective, the following diagnostic organizations cooperatively determined which services shall be available for implementation, as well as the process and method by which they shall be implemented:

- Vehicle Diagnostics,CoC, Electrical Distribution Systems Dept., DaimlerChrysler-Auburn Hills (DCA)
- Diagnostic Standards. EP/EID, DaimlerChrysler-Stuttgart (DCS)
- Electronics Design, Electrical/Electronic Systems Dept., Mitsubishi Motors Corporation (MMC)

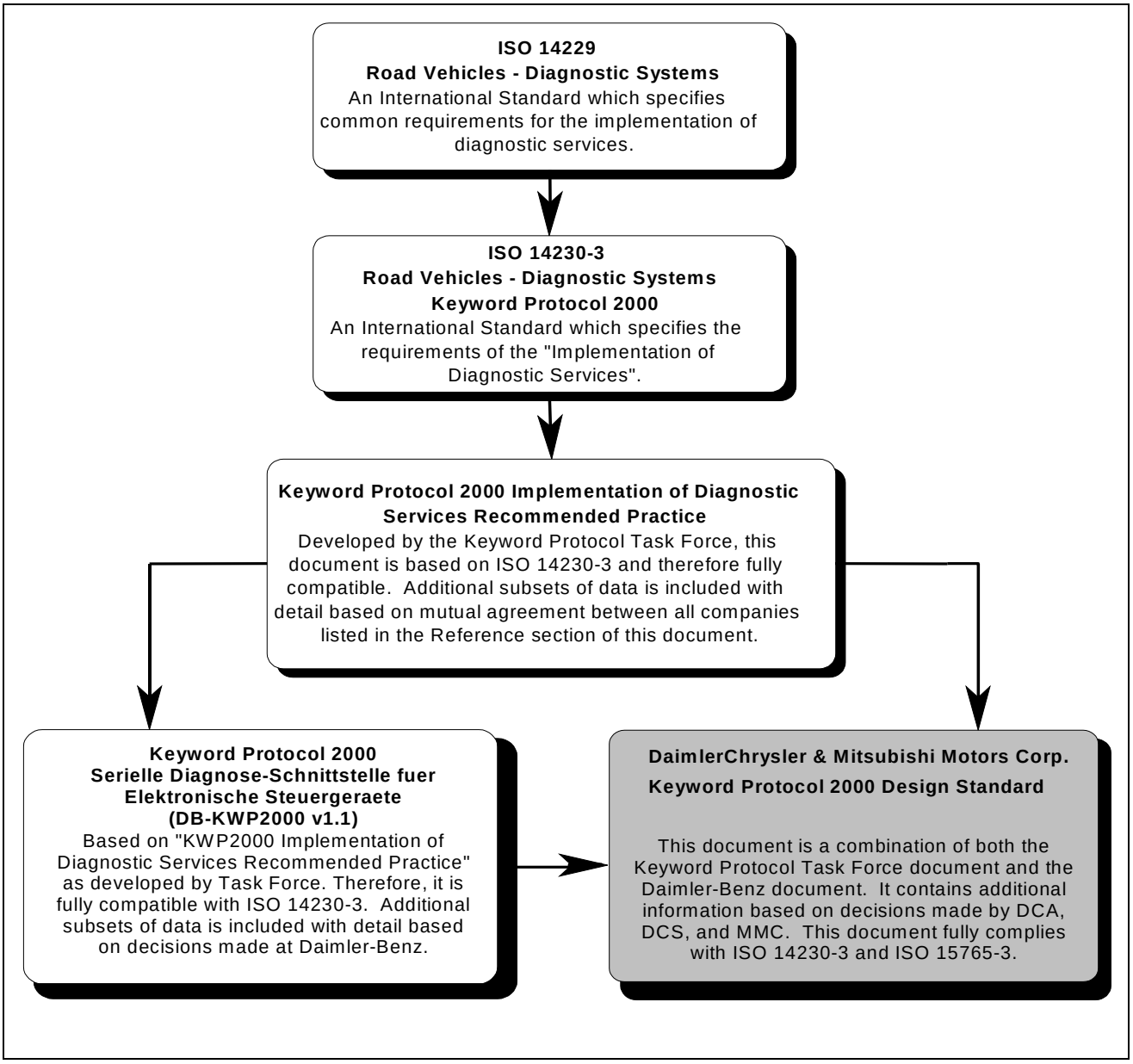

Figure 1 Hierarchy of Documents

2 GENERAL GUIDELINES FOR DIAGNOSTIC IMPLEMENTATION

2.1 COMMUNICATION BETWEEN A DIAGNOSTIC TOOL AND ECU

This section describes the message structure for request and response messages.

2.1.1 REQUEST MESSAGE FORMAT

A request message is used by the diagnostic tool to request an ECU to perform a specified diagnostic service.

Data Byte #	Data Value	Parameter Description	Message Usage
0	**$XX**	**Request Service ID**	**Mandatory**
1	**$XX** $YY : $ZZ	**Parameter #1** Valid Parameter #1 Value : Valid Parameter #1 Value	
2	**$XX** $YY : $ZZ	**Parameter #2** Valid Parameter #2 Value : Valid Parameter #2 Value	
:	**:**	**:**	
n	**$XX** $YY : $ZZ	**Parameter #m** Valid Parameter #m Value : Valid Parameter #m Value	

Table 2.1.1-1 Request Message Format Example

Column Description

Data Byte # – This defines the byte number for each parameter in the data stream. No data link or transport layer specific information is included.
Data Value – This defines the hexadecimal data value for each parameter in the data stream.
Parameter Description – This defines the purpose of each byte in the data stream. The byte description is in bold text. The options that are available for each parameter appear directly below the byte description.
Message Usage – This defines whether or not a particular parameter must be used in the data stream. The options are as follows:

- **Mandatory** – Each byte must appear in the message structure as specified. There are no exceptions.
- **Conditional** – Each byte can conditionally appear in the message structure based on the selection of other parameters passed within the message or based upon DCA, DCS, and MMC implementation requirements.
- **Optional** – Each byte can be optionally included in the message structure without an ECU or diagnostic tool "complaining" if the data is excluded.

2.1.2 POSITIVE RESPONSE MESSAGE FORMAT

A positive response may be transmitted by an ECU to indicate to a diagnostic test tool that a request message has been received. If an ECU is addressed by a diagnostic tool, it must return a positive response if it is able to completely perform the requested actions. The value returned by the ECU to indicate a positive response is determined as follows:

- Diagnostic Request Service ID Hex Value + $40

Data Byte #	Data Value	Parameter Description	Message Usage
0	Request Service ID + $40	Positive Response Service ID	Mandatory
1	$XX	Valid Parameter #1 Value	Conditional
:			
N	$XX	Valid Parameter #n-1 Value	Conditional

Table 2.1.2-1 Positive Response Message Format

For example, if a **Start Diagnostic Session** was requested, an ECU positive response data value is determined as follows:

- **Start Diagnostic Session Request ($10)+ $40 = Start Diagnostic Session Positive Response ($50)**

2.1.3 NEGATIVE RESPONSE MESSAGE FORMAT

A negative response shall be sent by an ECU if it is unable to perform the requested actions. The value returned by the ECU to indicate a negative response is always **$7F**. Within each Negative Response Message Format Table, under the parameter Negative Response Codes, resides a list of possible negative response codes. These Negative Response Codes are used to indicate the reason for the failed diagnostic action request.

Data Byte #	Data Value	Parameter Description	Message Usage
0	$7F	Negative Response Service ID	Mandatory
1	$XX	Request Service ID	Mandatory
2	$XX	Negative Response Code	Mandatory

Table 2.1.3-1 Negative Response Format

2.1.4 NO RESPONSE

There is no positive or negative response under the following conditions:

- If a diagnostic request message has explicitly selected the parameter to indicate that no response is required, there shall be no response (positive or negative) provided from the ECU. The services that currently support this sub-function are: **Disable Normal Message Transmission ($28)**, **Enable Normal Message Transmission ($29)**, **Tester Present ($3E)**, **Control DTC Setting ($85)** and **Response On Event ($86)**.

- If diagnostics within a bus system is initiated via a functionally addressed diagnostic request message for Start Diagnostic Session, there shall be no response (positive or negative) from the functionally addressed ECU's. Refer to Requirement 9 in Section 3.2, Service ID $10 – *Start Diagnostic Session* (Additional information pertaining to functional and physical addressing may be found in the "Diagnostic Performance Standard (DCA)" and in "Ausführungsvorschrift Diagnose: Diagnosespezifikation-Allgemeiner Teil" (DCS).)

3 DIAGNOSTIC SERVICE IDENTIFIERS

This section provides information pertaining to all of the DaimlerChrysler Corporation (DCX) and Mitsubishi Motors Corporation (MMC) supported diagnostic services. Diagnostic services, or service identifiers (SID's), are hexadecimal data values assigned to diagnostic messages, and allow for the exchange of diagnostic information between a diagnostic tool and an ECU. These diagnostic services may be used to monitor signals, interrogate data values, bi-directionally control I/O devices, read diagnostic trouble codes, clear diagnostic status information, or modify ECU behavior.

Keyword Protocol 2000 diagnostic services can be logically grouped into the following functional units:

- **Diagnostic Session Management** – This functional unit consists of Diagnostic Services that are used to initiate, terminate, configure, maintain, or alter diagnostic sessions. The services used to implement Diagnostic Session Management are as follows:
  - Start Diagnostic Session ($10)
  - ECU Reset ($11)
  - Read ECU Identification ($1A)
  - Security Access ($27)
  - Disable Normal Message Transmission ($28)
  - Enable Normal Message Transmission ($29)
  - Tester Present ($3E)
  - Control DTC Setting ($85)

- **Diagnostic Data Manipulation** – This functional unit consists of Diagnostic Services that allow an external diagnostic tool to send, receive, and alter data that is stored within an ECU. The services used to implement Diagnostic Data Manipulation are as follows:
  - Read Data By Local Identifier ($21)
  - Write Data By Local Identifier ($3B)
  - Read Data By Identifier ($22)
  - Write Data By Identifier ($2E)
  - Read Memory By Address ($23)
  - Write Memory By Address ($3D)
  - Read Diagnostic Trouble Codes By Status ($18)
  - Read Status of Diagnostic Trouble Codes ($17)
  - Clear Diagnostic Trouble Codes ($14)
  - Dynamically Define Local Identifiers ($2C)
  - Response On Event ($86)

- **Input / Output Control** – This functional unit consists of Diagnostic Services that allow an external diagnostic tool to control the states of I/O devices such as solenoids, relays, switches, or virtual devices in an ECU. The service used to implement Input / Output Control is as follows:
  - Input Output Control By Local Identifier ($30)

- **Remote Activation of Routine** - This functional unit consists of Diagnostic Services that allow an external diagnostic tool to start or stop a routine, as well as return the results of the routine. The services used to implement Remote Activation of Routines are as follows:
  - Start Routine By Local Identifier ($31)
  - Stop Routine By Local Identifier ($32)
  - Request Routine Results By Local Identifier ($33)

- **Upload / Download Control** - This functional unit consists of Diagnostic Services that allow an external diagnostic tool to request data transfer (download or upload) between an ECU and the external diagnostic tool. The services used to implement Upload / Download Control are as follows:
  - Request Download ($34)
  - Request Upload ($35)
  - Transfer Data ($36)
  - Request transfer Exit ($37)

3.1 DIAGNOSTIC SERVICE IDENTIFIERS AND PARAMETER SUMMARY TABLE

3.1.1 SUPPORTED DIAGNOSTIC SERVICES

Table 3.1.1-1 shows all of the KWP 2000 service identifiers defined within this document and which are supported by DaimlerChrysler/MMC. The list has been sorted according to the request ID assigned to each diagnostic service.

The first column lists the hexadecimal data value of the diagnostic service identifier (SID) for the request message. The second column lists a qualitative description of the diagnostic service identifier. The third column lists the page reference for a more detailed description of each service identifier and its respective parameters.

SID	Service Identifier Description	Page #
$10	*Start Diagnostic Session*	7
$11	*ECU Reset*	11
$14	*Clear Diagnostic Information*	12
$17	*Read Status Of Diagnostic Trouble Codes*	14
$18	*Read Diagnostic Trouble Codes By Status*	17
$1A	*Read ECU Identification*	23
$21	*Read Data By Local Identifier*	25
$22	*Read Data By Identifier*	27
$23	*Read Memory By Address*	29
$27	*Security Access*	31
$28	*Disable Normal Message Transmission*	35
$29	*Enable Normal Message Transmission*	37
$2C	*Dynamically Define Local Identifier*	38
$2E	*Write Data By Identifier*	44
$30	*Input Output Control By Local Identifier*	46
$31	*Start Routine By Local Identifier*	51
$32	*Stop Routine By Local Identifier*	53
$33	*Request Routine Results By Local Identifier*	55
$34	*Request Download*	56
$35	*Request Upload*	57
$36	*Transfer Data*	58
$37	*Request Transfer Exit*	60
$3B	*Write Data By Local Identifier*	64
$3D	*Write Memory By Address*	67
$3E	*Tester Present*	69
$85	*Control DTC Setting*	70
$86	*Response On Event*	72
$87 - $B9	*Reserved*	n/a
$BA - $BF	*System Supplier Specific*	n/a

Table 3.1.1-1 Service Identifier and Parameter Summary Table

3.2 SERVICE ID $10 – *START DIAGNOSTIC SESSION*

3.2.1 MESSAGE DESCRIPTION

PURPOSE

The service, **Start Diagnostic Session ($10)**, is used by the diagnostic tool to enable different types of diagnostic sessions in an ECU. In order to execute a diagnostic service the appropriate session has to be started first. See Table 3.2.1-1 on page 8 for a complete list of which service ID's are supported by each diagnostic session.

REQUIREMENTS

1. There shall be only one diagnostic session active at a time.
2. **Normal/Default Session ($81)** shall automatically be enabled by the ECU if no diagnostic session has been requested at power up but not if the ECU had entered an **ECU Passive Session ($90)** prior to power down.
3. An ECU shall return to **Normal/Default Session ($81)** after timeout of another diagnostic session.
4. An ECU shall be capable of providing all diagnostic functionality defined for the default diagnostic session under normal operating conditions.
5. The ECU shall first send a **Start Diagnostic Session Positive Response ($50)** service before the new session becomes active in the ECU.
6. A **Start Diagnostic Session Positive Response ($50)** service shall be returned by an ECU if the diagnostic tool requests a session that is already running (See Requirement 9), the ECU has already received the same request message previously and performed the requested operation, the ECU shall continue to perform the current operation (i.e. it is not a change of the session).
7. An ECU shall remain in its current diagnostic session if it is not able to switch into the requested diagnostic session.
8. The **Tester Present ($3E)** service shall be used to keep the following diagnostic sessions active if there are no other diagnostic messages sent within P3 max: **ECU Flash Reprogramming Session ($85)**, **Stand By Session ($89), Extended Diagnostic Session ($92)**.
9. If an ECU is functionally addressed with a Start Diagnostic Session ($10) request mesage, there shall be no response (positive or negative)

Table 3.2.1-1 lists all of the diagnostic service identifiers and the **Diagnostic Sessions** supported by this document. The 'X' indicates that a service identifier may be supported while the ECU is running in the specified **Diagnostic Session**. Each column indicates the available set of services that can be supported under each diagnostic session. Not all services are required under the selected session but should be implemented in order to achieve the desired functionality of the diagnostic session.

SID [1]	Service Identifier Description [2]	Normal / Default Session ($10 $81) [3]	ECU Flash Reprogramming Session ($10 $85) [4]	Stand By Session ($10 $89) [5]	ECU Passive Session ($10 $90) [6]	Extended Diagnostic Session ($10 $92) [7]	Page # [8]
$01-$09	SAE J1979 / ISO 15031-5 OBD II Test Sessions	X				X	-
$10	*Start Diagnostic Session*	X	X	X	X	X	7
$11	*ECU Reset*		X	X		X	11
$14	*Clear Diagnostic Information*					X	12
$17	*Read Status Of Diagnostic Trouble Codes*	X				X	14
$18	*Read Diagnostic Trouble Codes By Status*	X				X	17
$1A	*Read ECU Identification*	X	X			X	23
$21	*Read Data By Local Identifier*	X	X	X		X	25
$22	*Read Data By Identifier*	X	X	X		X	27
$23	*Read Memory By Address*					X	29
$27	*Security Access*		X	X		X	31
$28	*Disable Normal Message Transmission*					X	35
$29	*Enable Normal Message Transmission*					X	37
$2C	*Dynamically Define Local Identifier*					X	38
$2E	*Write Data By Identifier*		X	X		X	44
$30	*Input Output Control By Local Identifier*			X		X	46
$31	*Start Routine By Local Identifier*		X	X		X	51
$32	*Stop Routine By Local Identifier*		X	X		X	53
$33	*Request Routine Results By Local Identifier*		X	X		X	55
$34	*Request Download*		X			X	56
$35	*Request Upload*		X			X	57
$36	*Transfer Data*		X			X	58
$37	*Request Transfer Exit*		X			X	60
$3B	*Write Data By Local Identifier*		X	X		X	64
$3D	*Write Memory By Address*					X	67
$3E	*Tester Present*	X	X	X		X	69
$85	*Control DTC Setting*		X	X	X	X	70
$86	*Response On Event*	X				X	72
$BA - $BF	*System Supplier Specific*	(X)	(X)	(X)	(X)	(X)	n/a

Table 3.2.1-1 Service Identifier Session Support

COLUMN DEFINITIONS FOR TABLE 3.2.1-1

Column	Description
[1]	**SID** – This is the hexadecimal data value assigned to each service ID.
[2]	**Service Identifier Description** – The column is used for the qualitative description of the diagnostic service.
[3]	**Normal/Default Session ($81) –** See page 81 for definition**.**
[4]	**ECU Flash Reprogramming Session** – See page 81 for definition.
[5]	**Stand By Session** – See page 81 for definition.
[6]	**ECU Passive Session** – See page 81 for definition.
[7]	**Extended Diagnostic Session** – See page 81 for definition.
[8]	**Page # -** This column denotes the page on which additional information may be found describing the diagnostic service.

3.2.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$10**	**Start Diagnostic Session Request Service ID**	**Mandatory**	
1	**$XX**	**Diagnostic Session**	**Mandatory**	**81**
	$00-$7F	Reserved	N/A	
	$81	Normal/Default Session	Mandatory	81
	$85	ECU Flash Reprogramming Session	Conditional[1]	81
	$89	Stand By Session	Conditional[2]	81
	$90	ECU Passive Session	Conditional[2]	81
	$92	Extended Diagnostic Session	Mandatory	81
	$93 - $FF	Reserved	N/A	

Table 3.2.2-1 Start Diagnostic Session Request Message Format

Condition 1: **ECU Flash Reprogramming Session ($85)** shall be supported by each ECU which implements the ECU Flash Reprogramming Specification. (See Reference "*L"* in "Appendix A – *References*")
Condition 2: Required for all ECUs at DCA. For DCS refer to the respective model line specification.
Note: ECU Passive is required only as a developmental feature and shall be removed for production.

3.2.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$50**	**Start Diagnostic Session Positive Response Service ID**	**Mandatory**	
1	**$XX**	**Diagnostic Session** The **Diagnostic Session** parameter must be identical to the **Diagnostic Session** parameter sent in the request message. Refer to Table 3.2.2-1.	**Mandatory**	

Table 3.2.3-1 Start Diagnostic Session Positive Response Message Format

3.2.4 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Mandatory**	
1	**$10**	**Start Diagnostic Session Request Service ID**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.2.4-1 Start Diagnostic Session Negative Response Message Format

3.2.5 IMPLEMENTATION EXAMPLE OF **START DIAGNOSTIC SESSION (DIAGNOSTIC SESSION = EXTENDED DIAGNOSTIC SESSION**)

This section specifies the conditions to be fulfilled to successfully perform a Start Diagnostic Session in an ECU.

Hex	Diagnostic Tool Request Message
$10	**Start Diagnostic Session Request**
$92	Diagnostic Session = Extended Diagnostic Session

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$50	**Start Diagnostic Session Positive Response**	**$7F**	**Negative Response Service Identifier**
$92	Diagnostic Session = Extended Diagnostic Session	$10	Start Diagnostic Session Request
		$XX	Negative Response Code

Table 3.2.5-1 Implementation Example of Start Diagnostic Session

After completion of the **Start Diagnostic Session Positive Response ($10)** message, the **Extended Diagnostic Session ($92)** becomes active. If the ECU responds with a negative response code, the current diagnostic session shall remain active.

3.3 SERVICE ID $11 – *ECU RESET*

3.3.1 MESSAGE DESCRIPTION

PURPOSE

The service, **ECU Reset ($11)**, requests an ECU to effectively perform a reset based on the content of the **Reset Mode** parameter. The Reset Mode parameter may specify a reset for the entire ECU or selective portions of on-board memory.

REQUIREMENTS

1. An ECU shall send an **ECU Reset Positive Response ($51)** service before performing the specified Reset Mode.
2. After an ECU has performed a requested reset, the ECU shall return to **Normal/Default Diagnostic Session ($10 $81)**.

3.3.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$11**	**ECU Reset Request Service ID**	**Mandatory**	
1	**$XX**	**Reset Mode**	**Mandatory**	**82**
	$00	Reserved	N/A	
	$01	Power On Reset	Mandatory	82
	$82	Nonvolatile Memory Reset	Optional	82
	$02 - $FF	Reserved	N/A	

Table 3.3.2-1 ECU Reset Request Message Format

3.3.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$51**	**ECU Reset Positive Response Service ID**	**Mandatory**	

Table 3.3.3-1 ECU Reset Positive Response Message Format

3.3.4 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Mandatory**	
1	**$11**	**ECU Reset Request Service ID**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.3.4-1 ECU Reset Negative Response Message

3.3.5 IMPLEMENTATION EXAMPLE OF **ECU RESET** (RESET MODE = POWER ON)

This section specifies the conditions to be fulfilled to successfully perform an **ECU Reset**.

Hex	Diagnostic Tool Request Message
$11	**ECU Reset Request**
$01	Reset Mode = Power On

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$51	**ECU Reset Positive Response**	**$7F**	**Negative Response Service Identifier**
		$11	ECU Reset Request
		$XX	Negative Response Code

Table 3.3.5-1 Implementation Example of ECU Reset

After the ECU has sent the **Reset Positive Response ($11)** message, the ECU shall reset and return to **Normal/Default Session ($81)**.

3.4 SERVICE ID $14 – *CLEAR DIAGNOSTIC INFORMATION*

3.4.1 MESSAGE DESCRIPTION

PURPOSE

The service, **Clear Diagnostic Information ($14),** is used by the diagnostic tool to clear **Diagnostic Trouble Codes** (**DTC**) and associated diagnostic information in the ECU's memory.

Note for Implementation at DCA: A complete list of **Diagnostic Trouble Codes** for Powertrain (P), Body (B), Chassis (C), and Network (U) will be provided by Vehicle Diagnostics, CoC.

REQUIREMENTS

1. If no DTC and/or diagnostic information was stored, the ECU shall return the **Clear Diagnostic Information Positive Response ($54)** and the Group of DTC parameters after receiving the Clear **Diagnostic Information Request ($14)** service

The diagnostic tool shall send the Group Of DTC parameter in the **Clear Diagnostic Information Request ($14)** service to specify a single DTC or a group of DTC's to be cleared. A value other than $0000, $4000, $8000, $C000, or $FF00 implies a single DTC.

A successful completion of Clear Diagnostic Information shall reset all DTC related information in the ECU's memory specified by the Group Of DTC parameter. (Note: For DCA, this excludes data stored in the Fault Historical Record / Diagnostic Tell – Tale Retention Stack.)

DCA Only: All DTC values used in the Group Of DTC parameter must be coordinated with the Vehicle Diagnostics, CoC group.

3.4.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$14**	**Clear Diagnostic Information Request Service ID**	**Mandatory**	
1	**$XX**	**Group Of DTC (High Byte)**	**Mandatory**	**82**
2	**$XX**	**Group Of DTC (Low Byte)**	**Mandatory**	**82**
	$0000	All Powertrain DTC's	Optional	82
	$0001 - $3FFF	Powertrain DTC	Optional	83
	$4000	All Chassis DTC's	Optional	82
	$4001 - $7FFF	Chassis DTC	Optional	83
	$8000	All Body DTC's	Optional	82
	$8001 - $BFFF	Body DTC	Optional	83
	$C000	All Network DTC's	Optional	82
	$C001 - $FEFF	Network DTC	Optional	83
	$FF00	All DTC's	Mandatory	83

Table 3.4.2-1 Clear Diagnostic Information Request Message Format

3.4.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$54**	**Clear Diagnostic Information Positive Response Service ID**	**Mandatory**	
1	**$XX**	**Group Of DTC ((High Byte)**	**Mandatory**	**82**
2	**$XX**	**Group Of DTC (Low Byte)**	**Mandatory**	**82**
		The **Group Of DTC** parameters must be identical to the **Group Of DTC** parameters sent in the request message. Refer to Table 3.4.2-1		

Table 3.4.3-1 Clear Diagnostic Information Positive Response Message Format

3.4.4 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Mandatory**	
1	**$14**	**Clear Diagnostic Information Request Service**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.4.4-1 Clear Diagnostic Information Negative Response Message

3.4.5 IMPLEMENTATION EXAMPLE OF **CLEAR DIAGNOSTIC INFORMATION** (FUNCTION GROUP)

This section specifies the conditions to be fulfilled to successfully perform a **Clear Diagnostic Session** in the ECU. In this example, the **Group Of DTC** parameter is set to "**Request By Function = Powertrain"** in order to erase all **DTC's** in the Powertrain functional group. Prior to the clearing of the Diagnostic Information by functional group, the ECU had stored 2 DTC's:

- P0130 O2 Sensor Circuit Malfunction (Bank1, Sensor 1);
- P0120 Throttle Position Circuit Malfunction

Hex	Diagnostic Tool Request Message
$14	**Clear Diagnostic Information Request**
$00	Group Of DTC – Powertrain (High Byte)
$00	Group Of DTC – Powertrain (Low Byte)

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$54	**Clear Diagnostic Information Positive Response**	**$7F**	**Negative Response Service Identifier**
$00	Group Of DTC – Powertrain (High Byte)	$14	Clear Diagnostic Information Request
$00	Group Of DTC – Powertrain (Low Byte)	$XX	Negative Response Code

Table 3.4.5-1 Implementation Example of Clear Diagnostic Information By Function Group

After the ECU responds with the **Clear Diagnostic Information Positive Response ($54)**, both of the DTC's listed are erased from the Powertrain functional group.

3.4.6 IMPLEMENTATION EXAMPLE OF **CLEAR DIAGNOSTIC INFORMATION** (DTC)

In this example, **Group Of DTC** is set to "**Request By DTC = $0120**" in order to erase a specific **DTC** in Powertrain functional group. Prior to the clearing of the Diagnostic Information by DTC, the ECU had stored 2 DTC's:

- P0130 O2 Sensor Circuit Malfunction (Bank1, Sensor 1);
- P0120 Throttle Position Circuit Malfunction

Hex	Diagnostic Tool Request Message
$14	**Clear Diagnostic Information Request**
$01	Group Of DTC – DTC P0120 (High Byte)
$20	Group Of DTC – DTC P0120 (Low Byte)

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$54	**Clear Diagnostic Information Positive Response**	**$7F**	**Negative Response Service Identifier**
$01	Group Of DTC – DTC P0120 (High Byte)	$14	Clear Diagnostic Information Request
$20	Group Of DTC – DTC P0120 (Low Byte)	$XX	Negative Response Code

Table 3.4.6-1 Implementation Example of Clear Diagnostic Information By DTC

After the ECU responds with the **Clear Diagnostic Information Positive Response ($54)**, the DTC=$0120 is erased. The DTC=$0130 remains unaffected in this example.

3.5 SERVICE ID $17 – *READ STATUS OF DIAGNOSTIC TROUBLE CODES*

3.5.1 MESSAGE DESCRIPTION

PURPOSE

The service, **Read Status of Diagnostic Trouble Codes ($17),** is used to read DTC information. This service is intended to acquire data about a single DTC per request and response. The reason for this limitation is various ECU's support many different DTC's and system supplier data.

Note for Implementation at DCA: A complete list of **Diagnostic Trouble Codes** for Powertrain (P), Body (B), Chassis (C), and Network (U) will be provided by Vehicle Diagnostics, CoC.

REQUIREMENTS

1. The **Read Status Of Diagnostic Trouble Codes Negative Response ($7F)** service shall always return an appropriate negative response code when the **Read Status Of Diagnostic Trouble Codes ($17)** service cannot be fulfilled.
2. The ECU shall send a **Read Status Of Diagnostic Trouble Codes Negative Response ($7F)** service with the appropriate negative response code if the diagnostic tool requests the status of a DTC which is unknown to the ECU.
3. The number of System Supplier Data may vary for each DTC.
4. An ECU shall set the Number Of DTC Stored parameter to "$00" if it does not have any DTC status information stored. Since no DTC's are stored, no additional info (i.e. DTC and Status of DTC) is reported in **Read Status Of Diagnostic Trouble Codes Positive Response ($57)** service.
5. DCA Only: All DTC values used in the **Diagnostic Trouble Code** parameter must be coordinated with Vehicle Diagnostics, CoC.

3.5.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$17**	**Read Status Of Diagnostic Trouble Codes Request Service ID**	**Mandatory**	
1	**$XX**	**Diagnostic Trouble Code { High Byte }**	**Mandatory**	**83**
2	**$XX**	**Diagnostic Trouble Code { Low Byte }**	**Mandatory**	**83**
	$0001 - $3FFF	Powertrain DTC	Conditional[1]	83
	$4001 - $7FFF	Chassis DTC	Conditional[2]	83
	$8001 - $BFFF	Body DTC	Conditional[3]	83
	$C001 - $FEFF	Network DTC	Conditional[4]	83

Table 3.5.2-1 Read Status Of Diagnostic Trouble Codes Request Message Format

Condition[1]: Powertrain DTCs shall be supported by ECUs capable of detecting powertrain related failures.
Condition[2]: Chassis DTCs shall be supported by ECUs capable of detecting chassis related failures.
Condition[3]: Body DTCs shall be supported by ECUs capable of detecting body related failures.
Condition[4]: Network DTCs shall be supported by ECUs capable of detecting network(bus) related failures.

3.5.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$57**	**Read Status Of Diagnostic Trouble Codes Positive Response Service ID**	**Mandatory**	
1	**$01**	**Number OF DTC**	**Mandatory**	**83**
2	**$XX**	**Diagnostic Trouble Code { High Byte }**	**Conditional[1]**	**83**
3	**$XX**	**Diagnostic Trouble Code { Low Byte }** The **Diagnostic Trouble Code** parameters must be identical to the DTC specified **Diagnostic Trouble Code** sent in the request message. Refer to Table 3.5.2-1	**Conditional[1]**	**83**
4	**$XX**	**Status Of DTC**	**Conditional[1]**	**84**
5	**$XX**	**System Supplier Data #1**	**Conditional[2]**	**85**
:	**:**	**:**		
n	**$XX**	**System Supplier Data #m**	**Conditional[2]**	**85**

Table 3.5.3-1 Read Status Of Diagnostic Trouble Codes Positive Response Message Format

Condition[1]: The DTC and Status of DTC shall be returned if the Number of DTC=$01. If the Number of DTC=$00, the DTC and Status of DTC shall not be included in the Positive Response.
Condition[2]: System Supplier Data shall be supported if additional information for the DTC is stored and the Number of DTC=$01.

3.5.4 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Mandatory**	
1	**$17**	**Read Status Of Diagnostic Trouble Codes Request Service ID**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.5.4-1 Read Status Of Diagnostic Trouble Codes Negative Response Message

3.5.5 IMPLEMENTATION EXAMPLE OF **READ STATUS OF DIAGNOSTIC TROUBLE CODES** (GROUP OF DTC=$0120)

Hex	Diagnostic Tool Request Message
$17	**Read Status Of DTC Request**
$01	Request by DTC [High Byte] {Throttle Position Malfunction}
$20	Request by DTC [Low Byte] {Throttle Position Malfunction}

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$57	**Read Status Of DTC Positive Response**	**$7F**	**Negative Response Service Identifier**
$01	Number Of DTC	$17	Read Status Of DTC Request
$01	DTC [High Byte] {Throttle Position Malfunction}	$XX	Negative Response Code
$20	DTC [Low Byte] {Throttle Position Malfunction}		
$E0	Status Of DTC		
$09	System Supplier Data #1 {Odometer (MSB)}		
$92	System Supplier Data #1 {Odometer (LSB)} (4900 miles)		
$00	System Supplier Data #2 {Accumulation Timer (MSB)}		
$07	System Supplier Data #3 {Accumulation Timer (LSB)}		
	(7 min)		
$20	System Supplier Data #3 {Ignition Counter = 32 counts}]		

Table 3.5.5-1 Implementation Example of Group Of DTC

In this example, the **Read Status Of DTC Positive Response ($57)** returns a value of "$E0" for the **Status Of DTC**. This value is determined as follows:

Status Of DTC = $E0 = **11100000b**

DTC Fault Symptom = **'0000b'** {Reserved}
DTC Test Complete = **'0b'** {test complete}
DTC Storage State = **'11b'** {active/stored DTC}
DTC Warning Lamp = **'1b'** { on }

Refer to section 4.3.4for a complete description of the bit level implementation of the parameter **Status Of DTC**.

3.6 SERVICE ID $18 – *READ DIAGNOSTIC TROUBLE CODES BY STATUS*

3.6.1 MESSAGE DESCRIPTION

PURPOSE

The service, **Read Diagnostic Trouble Codes By Status ($18),** is used by the diagnostic tool to read **DTC's** by status from the ECU's memory.

Note for Implementation at DCA: A complete list of **Diagnostic Trouble Codes** for Powertrain (P), Body (B), Chassis (C), and Network (U) will be provided by Vehicle Diagnostics, CoC.

REQUIREMENTS

1. The **Read Diagnostic Trouble Codes By Status Positive Response ($58)** service shall always return the Number Of DTCs, DTC Value, and Status Of DTC parameters for a group of DTC's as specified in the **Read Diagnostic Trouble Codes By Status Request ($18)** service.
2. The **Read Diagnostic Trouble Codes By Status Negative Response ($7F)** service shall always return an appropriate negative response code when the **Read Diagnostic Trouble Codes By Status Request ($18)** service cannot be fulfilled.
3. An ECU shall set the Number Of DTCs Stored parameter to "$00" if it does not have any DTC status information stored. Since no DTC's are stored, no additional info is reported in the **Read Status Of Diagnostic Trouble Codes Positive Response ($58)** service.
4. Within one positive response each DTC can be reported only once
5. DCA Only: All DTC values used in the Group Of DTC parameter must be coordinated with the Vehicle Diagnostics,CoC group.
6. All DTC's returned in the positive response shall be returned in chronolgical order from newest (most recent) to oldest.
7. When using the Request Extended Number of Supported DTCs ($E0), the service **Request Supported DTC and Status ($18 $01 / $18 $03)** have to be repeated within a certain time window (e.g. P3-max). After the window has elapsed the ECU starts to report the DTCs from the beginning again, if the service **Request Supported DTC and Status ($18 $01 / $18 $03)** is requested again.

3.6.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$18**	**Read Diagnostic Trouble Codes By Status Request Service ID**	**Mandatory**	
1	**$XX**	**Status Of DTC**	**Mandatory**	**83**
	$00	Request Identified DTC and Status (SAE J2012 / ISO 15031-6 format)	Conditional[1]	83
	$01	Request Supported DTC and Status (SAE J2012 / ISO 15031-6 format)	Conditional[3]	83
	$02	Request Identified 2 Byte Hex DTC and Status	Conditional[2]	83
	$03	Request Supported 2 Byte Hex DTC and Status	Conditional[3]	84
	$04	Request Most Recent DTC	Optional	84
	$E0	Request Extended Number of Supported DTCs	Conditional[4]	84
2	**$XX**	**Group Of DTC = Request By Functional Group (High Byte)**	**Mandatory**	**82**
3	**$XX**	**Group Of DTC = Request By Functional Group (Low Byte)**	**Mandatory**	**82**
	$0000	All Powertrain DTC's	Optional	82
	$4000	All Chassis DTC's	Optional	82
	$8000	All Body DTC's	Optional	82
	$C000	All Network DTC's	Optional	82
	$FF00	All DTC's	Mandatory	83

Table 3.6.2-1 Read Diagnostic Codes By Status Request Message Format

Condition[1]: This sub-function shall be supported by all DCA and MMC ECUs.
Condition[2]: This sub-function shall be supported by all DCS-ECUs.
Condition[3]: This sub-function shall be supported if the DTC Readiness Flag information shall be supported.
Condition[4]: This sub-function shall be supported if the number of supported DTCs exceeds 255 or if the ECU requires segmentation on application layer to transmit all supported DTCs.

3.6.3 POSITIVE RESPONSE MESSAGE FORMAT (FOR REQUEST PARAMETERS 00-04)

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$58**	**Read Diagnostic Trouble Codes By Status Positive Response Service ID**	**Mandatory**	
1	**$XX**	**Number OF DTCs**	**Mandatory**	**83**
2	**$XX**	**Diagnostic Trouble Code #1 {High Byte }**	**Mandatory**	**83**
3	**$XX**	**Diagnostic Trouble Code #1 {Low Byte }**	**Mandatory**	**83**
4	**$XX**	**Status Of DTC#1**	**Mandatory**	**84**
:	**:**	**:**	**:**	**:**
n-2	**$XX**	**Diagnostic Trouble Code #m {High Byte }**	**Conditional**	**83**
n-1	**$XX**	**Diagnostic Trouble Code #m {Low Byte }**	**Conditional**	**83**
n	**$XX**	**Status Of DTC #m**	**Conditional**	**84**

Table 3.6.3-1 Read Diagnostic Codes By Status Positive Response Message Format

Condition: Bytes n-2 to n are to be returned by the ECU if more than one DTC has been requested.

3.6.4 POSITIVE RESPONSE MESSAGE FORMAT ($E0)

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$58**	**Read Diagnostic Trouble Codes By Status Positive Response Service ID**	**Mandatory**	
1	**$XX**	**Extended Number OF DTCs {High Byte }**	**Mandatory**	**83**
2	**$XX**	**Extended Number OF DTCs {Low Byte }**	**Mandatory**	**83**

Table 3.6.4-1 Read Diagnostic Codes By Status – Extended Number of DTCs Positive Response Message Format

3.6.5 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Mandatory**	
1	**$18**	**Read Status Of Diagnostic Trouble Codes Request Service ID**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.6.5-1 Read Diagnostic Codes By Status Positive Response Message Format

3.6.6 IMPLEMENTATION EXAMPLE OF **READ DIAGNOSTIC TROUBLE CODES BY STATUS** (STATUS OF DTC =$00 AND GROUP OF DTC = $FF00)

This example begins with an ECU having the following DTC's information stored in memory:

DTC #1 : P0130: O2 Sensor Circuit Malfunction (Bank 1, Sensor 1)
Status Of DTC #1 = $20 = 00100000b

DTC Fault Symptom = **'0000b'** {Reserved}
DTC Readiness Flag = **'0b'** {Test Complete}
DTC Storage State = **'01b'** {Not Present / stored}
Warning Indicator Request State = **'0b'** {Off}

DTC #2: P0120: Throttle Position Malfunction
Status Of DTC #2 = $E0= 11100000b

DTC Fault Symptom = **'0000b'** {Reserved}
DTC Readiness Flag = **'0b'** {Test Complete}
DTC Storage State = **'11b'** {active/stored}
Warning Indicator Request State = **'1b'** {on}

DTC #3: P0135: O2 Sensor Heater Circuit Malfunction (Bank 1, Sensor 1)
Status Of DTC #3 = $10 = 00010000b

DTC Fault Symptom = **'0000b'** {Reserved}
DTC Readiness Flag = **'1b'** {Test Not Complete}
DTC Storage State = **'00b'** {No DTC Detected}

Warning Indicator Request State = **'0b'** {off}

DTC#4: U0101: Lost Communication with TCM
Status Of DTC #4 = $60 = 01100000b

DTC Fault Symptom = **'0000b'** {Reserved}
DTC Readiness Flag = **'0b'** {Test Complete}
DTC Storage State = **'11b'** {active/stored}
Warning Indicator Request State = '0b' **{off}**

The diagnostic tool is requesting the ECU to return All DTC's.

Hex	Diagnostic Tool Request Message
$18	**Read DTC By Status Request**
$00	Status Of DTC { Request Identified DTC and Status (SAE J2012 / ISO 15031-6 format)}
$FF	Group Of DTC [High Byte] {All DTC's}
$00	Group Of DTC [Low Byte] {All DTC's}

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$58	**Read DTC By Status Positive Response**	**$7F**	**Negative Response Service Identifier**
$03	Number Of DTCs	$18	Read DTC By Status Positive Response
$01	DTC #1 [High Byte]{O2 Sensor Circuit Malfunction} (P0130)	$XX	Negative Response Code
$30	DTC #1 [Low Byte] {Bank 1, Sensor 1}		
$20	Status Of DTC #1		
$01	DTC #2 [High Byte]{Throttle Position Malfunction} (P0120)		
$20	DTC #2 [Low Byte] {Throttle Position Malfunction}		
$E0	Status Of DTC #2		
$C1	DTC #4 [High Byte]{Lost Communication with TCM} (U0101)		
$01	DTC #4 [Low Byte] {Lost Communication with TCM}		
$60	Status Of DTC #4		

Table 3.6.6-1 Implementation Example #1 of Status of DTC

Note: DTC #3 is not included in the **Read DTC By Status Positive Response ($58)** message because the **DTC Storage State** is still '**00b**', No DTC Detected

3.6.7 IMPLEMENTATION EXAMPLE OF **READ DIAGNOSTIC TROUBLE CODES BY STATUS** (STATUS OF DTC =$02 AND GROUP OF DTC = $0000)

Note: The DTC information stored in memory is as follows:
DTC #1 : 2005: Engine Temperature
Status Of DTC #1 = $20 = 00100000b

DTC Fault Symptom = **'0000b'** {Reserved}
DTC Readiness Flag = **'0b'** {Test Complete}
DTC Storage State = **'01b'** {Not Present / stored}
Warning Indicator Request State = **'0b'** {Off}

DTC #2: 203F: Oil Quality
Status Of DTC #2 = $E0= 11100000b

DTC Fault Symptom = **'0000b'** {Reserved}
DTC Readiness Flag = **'0b'** {Test Complete}
DTC Storage State = **'11b'** {active/stored}
Warning Indicator Request State = **'1b'** {on}

DTC #3: 2135: O2 Sensor Heater Circuit Malfunction (Bank 1, Sensor 1)
Status Of DTC #3 = $10 = 00010000b

DTC Fault Symptom = **'0000b'** {Reserved}
DTC Readiness Flag = **'1b'** {Test Not Complete}
DTC Storage State = **'00b'** {No DTC Detected}

Warning Indicator Request State = **'0b'** {off}

Hex	Diagnostic Tool Request Message
$18	**Read DTC By Status Request**
$02	Status Of DTC {Request Identified DTC And Status}
$00	Group Of DTC [High Byte] {All Powertrain DTC's}
$00	Group Of DTC [Low Byte] {All Powertrain DTC's}

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$58	**Read DTC By Status Positive Response**	**$7F**	**Negative Response Service Identifier**
$02	Number Of DTCs	$18	Read DTC By Status Positive Response
$20	DTC #1 [High Byte] { Engine Temperature }	$XX	Negative Response Code
$05	DTC #1 [Low Byte] { Engine Temperature }		
$20	Status Of DTC #1		
$20	DTC #2 [High Byte] { Oil Quality }		
$3F	DTC #2 [Low Byte] { Oil Quality }		
$E0	Status Of DTC #2		

Table 3.6.7-1 Implementation Example #1 of Status of DTC

Note: DTC #3 is not included in the **Read DTC By Status Positive Response ($58)** message because the **DTC Storage State** is still '**00b**', No DTC Detected.

3.6.8 IMPLEMENTATION EXAMPLE OF **READ DIAGNOSTIC TROUBLE CODES BY STATUS** (STATUS OF DTC = $03 AND GROUP OF DTC = $0000)

In this example, the diagnostic tool is requesting the ECU return all supported DTC's in the Powertrain functional group regardless of the status of the **DTC Readiness Flag**. This is done by setting the **Status OF DTC** to $03 and setting the **Group Of DTC** to $0000 in the **Read Diagnostic Trouble Codes By Status Request ($18)** message.

Note: The DTC information stored in memory is identical to that in the example illustrated in Section 3.6.6.

Hex	Diagnostic Tool Request Message
$18	**Read DTC By Status Request**
$03	Status Of DTC { Request Supported DTC And Status }
$00	Group Of DTC [High Byte] {Powertrain}
$00	Group Of DTC [Low Byte] {Powertrain}

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$58	**Read DTC By Status Positive Response**	**$7F**	**Negative Response Service Identifier**
$03	Number Of DTCs	$18	Read DTC By Status Request
$20	DTC #1 [High Byte] { Engine Temperature }	$XX	Negative Response Code
$05	DTC #1 [Low Byte] { Engine Temperature }		
$20	Status Of DTC #1		
$20	DTC #2 [High Byte] { Oil Quality }		
$3F	DTC #2 [Low Byte] { Oil Quality }		
$E0	Status Of DTC #2		
$20	DTC #3 [High Byte] {O2 Sensor Heater Circuit Malfunction}		
$35	DTC #3 [Low Byte] { Bank 1, Sensor 1 }		
$10	Status Of DTC #3		

Table 3.6.8-1 Implementation Example #2 of Status of DTC

3.6.9 IMPLEMENTATION EXAMPLE OF **READ DIAGNOSTIC TROUBLE CODES BY STATUS** (STATUS OF DTC = $03 AND GROUP OF DTC = $FF00) WITH EXTENDED NUMBER OF DTCS

In this example, the diagnostic tool is requesting the ECU return all supported DTC's regardless of the status of the **DTC Readiness Flag**. This is done by setting the **Status OF DTC=**$03 (or **Status of DTC**=$01) and setting the **Group Of DTC=**$FF00 in the **Read Diagnostic Trouble Codes By Status Request ($18)** message. Because the ECU supports more than 255 DTCs (known offboard) the extended method has to be used.

First the total number of supported DTCs is read.

Hex	Diagnostic Tool Request Message
$18	**Read DTC By Status Request**
$E0	Status Of DTC { Request Supported DTC And Status }
$FF	Group Of DTC [High Byte] {all DTCs}
$00	Group Of DTC [Low Byte] {all DTCs}

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$58	**Read DTC By Status Positive Response**	**$7F**	**Negative Response Service Identifier**
$01	Number Of DTCs [High Byte]	$18	Read DTC By Status Request
$05	Number Of DTCs [Low Byte]	$XX	Negative Response Code

Table 3.6.9-1 Implementation Example #3 of Status of DTC

Next, the Service $18 $03 is repeated until all DTCs are read. The services have to be repeated within a certain time window. After the window has elapsed the ECU starts to report the DTCs from the beginning again when requested.

Hex	Diagnostic Tool Request Message
$18	**Read DTC By Status Request**
$03	Status Of DTC { Request Supported DTC And Status }
$FF	Group Of DTC [High Byte] {all DTCs}
$00	Group Of DTC [Low Byte] {Powertrain}

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$58	**Read DTC By Status Positive Response**	**$7F**	**Negative Response Service Identifier**
$FF	Number Of DTCs	$18	Read DTC By Status Request
$Ax	DTC #1 [High Byte]	$XX	Negative Response Code
$Ay	DTC #1 [Low Byte]		
$Az	Status Of DTC #1		
:			
$Bx	DTC #255 [High Byte]		
$By	DTC #255 [Low Byte]		
$Bz	Status Of DTC #255		

Table 3.6.9-2 Implementation Example #3 of Status of DTC

Hex	Diagnostic Tool Request Message
$18	**Read DTC By Status Request**
$03	Status Of DTC { Request Supported DTC And Status }
$FF	Group Of DTC [High Byte] {all DTCs}
$00	Group Of DTC [Low Byte] {Powertrain}

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$58	**Read DTC By Status Positive Response**	**$7F**	**Negative Response Service Identifier**
$06	Number Of DTCs	$18	Read DTC By Status Request
$Ax	DTC #256 [High Byte]	$XX	Negative Response Code
$Ay	DTC #256 [Low Byte]		
$Az	Status Of DTC #256		
:			
$Bx	DTC #261 [High Byte]		
$By	DTC #261 [Low Byte]		
$Bz	Status Of DTC #261		

Table 3.6.9-3 Implementation Example #3 of Status of DTC

3.7 SERVICE ID $1A – *READ ECU IDENTIFICATION*

3.7.1 MESSAGE DESCRIPTION

PURPOSE

The service, **Read ECU Identification ($1A),** requests identification data from an ECU. The type of identification data requested by the diagnostic tool is identified by the parameter **Identification Option**. The Identification Option may provide the ECU part number, serial number, software revision level, model year, etc. The **Identification Option** is written by using the **Write Data By Local Identifier ($3B)** service when applicable.

3.7.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$1A**	**Read ECU Identification Request Service ID**	**Mandatory**	
1	**$XX**	**Local Identifier = Identification Option**	**Mandatory**	**86**
	$80 - $85	Reserved	N/A	
	$86	DCS ECU Identification	Conditional[1]	86
	$87	DCX / MMC ECU Identification (Note: DCX = DCA + DCS)	Mandatory	87
	$88	VIN (Orginal)	Conditional[2]	88
	$89	Diagnostic Variant Code	Conditional[3]	89
	$8A - $8F	System Supplier Specific	Optional	105
	$90	VIN (Current)	Conditional[2]	89
	$96	Calibration Identification	Optional	89
	$97	Calibration Verification Number	Optional	90
	$9A	ECU Code Fingerprint	Conditional[4]	90
	$9B	ECU Data Fingerprint	Conditional[4]	91
	$9C	ECU Code Software Identification	Conditional[4]	91
	$9D	ECU Data Software Identification	Conditional[4]	92
	$9E	ECU Boot Software Identification	Conditional[4]	92
	$9F	ECU Boot Fingerprint	Conditional[4]	92

Table 3.7.2-1 Read ECU Identification Request Message Format

Condition[1]: This Identification Option shall be supported by ECUs that are used at DCS.
Condition[2]: This Identification Option shall be supported by ECUs that are used at DCA but is optional for ECU's used at DCS.
Condition[3]: This Identification Option shall be supported by ECUs that are used at MMC.
Condition[4]: This Identification Option shall be supported according to the Flash Reprogramming Requirements Specification. (See Reference "*L"* in "Appendix A – *References*")

3.7.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$5A**	**Read ECU Identification Positive Response Service ID**	**Mandatory**	
1	**$XX**	**Local Identifier = Identification Option** The **Identification Option** parameter must be identical to the **Identification Option** parameter sent in the request message. Refer to Table 3.7.2-1	**Mandatory**	**86**
2	**$XX**	**ECU Identification Parameter#1**	**Mandatory**	**85**
:	**:**	**:**		
n	**$XX**	**ECU Identification Parameter #m**	Conditional	**85**

Table 3.7.3-1 Read ECU Identification Positive Response Message Format

Condition: ECU Identification Parameter #m shall be reported if defined.

3.7.4 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Mandatory**	
1	**$1A**	**Read ECU Identification Request Service ID**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.7.4-1 Read ECU Identification Positive Response Message Format

3.7.5 IMPLEMENTATION EXAMPLE OF **READ ECU IDENTIFICATION** (IDENTIFICATION OPTION = DCA ECU IDENTIFICATION)

This example shows a **Read ECU Identification Request ($1A)** message which is sent to the ECU with the **Identification Option** = **DCX ECU Identification ($87)**.

Hex	Diagnostic Tool Request Message
$1A	**Read ECU Identification Request**
$87	Identification Option = DCX ECU Identification

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$5A	**Read ECU Identification Positive Response**	**$7F**	**Negative Response Service Identifier**
$87	Identification Option = DCX ECU Identification	$1A	Read ECU Identification Request
$02	ECU Origin: **DCA**	$XX	Negative Response Code
$12	Supplier Identification		
$FF	Diagnostic Information (High Byte) – Refer to Config		
$0A	Diagnostic Information (Middle Byte) – Revision 10		
$FF	Reserved		
$01	Hardware Version (Major Byte)		
$05	Hardware Version (Minor Byte)		
$02	Software Version (High Byte)		
$03	Software Version (Middle Byte)		
$00	Software Version (Low Byte)		
$35	Part Number Position #1 (High Byte) [5]		
$31	Part Number Position #2 [1]		
$38	Part Number Position #3 [8]		
$37	Part Number Position #4 [7]		
$33	Part Number Position #5 [3]		
$33	Part Number Position #6 [3]		
$30	Part Number Position #7 [0]		
$32	Part Number Position #8 [2]		
$41	Part Number Position #9 [A]		
$42	Part Number Position #10 (Low Byte) [B]		

Table 3.7.5-1 Implementation Example of Read ECU Identification

3.8 SERVICE ID $21 – *READ DATA BY LOCAL IDENTIFIER*

3.8.1 MESSAGE DESCRIPTION

PURPOSE

The service, **Read Data By Local Identifier ($21),** requests blocks of data from the ECU identified by the **Record Local Identifier**. The blocks of data read may include sensor information, actuator status, and may be of varying length. The data is written to the same **Local Identifier** by using the **Write Data By Local Identifier ($3B)** service.

REQUIREMENTS

Reading DCS Variant Coding shall be performed with **Read Data By Local Identifier ($21).**

3.8.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$21**	**Read Data By Local Identifier Request Service ID**	**Mandatory**	
1	**$XX**	**Record Local Identifier**	**Mandatory**	**85**
	$00	Reserved	N/A	
	$01-$7F	Record Local Identifier	Optional	85
	$80-$9F	This range of values is reserved for use by the service **Read ECU Identification ($1A**). DO NOT READ VALUES IN THIS RANGE WITH **READ DATA BY LOCAL IDENTIFIER ($21).**	N/A	
	$A0-$DF	Record Local Identifier	Optional	85
	$E0	Development Data	Conditional[1]	92
	$E1	ECU Serial Number	Mandatory	93
	$E2	DBCom Data	Optional	93
	$E3	Operating System Version	Optional	94
	$E4	ECU Reprogramming Identification	Conditional[5]	94
	$E5	Vehicle Information	Conditional[2]	94
	$E6	Flash Info 1	Conditional[3]	95
	$E7	Flash Info 2	Conditional[3]	95
	$E8	System Diagnostic general parameter data	Conditional[4]	95
	$E9	System Diagnostic global parameter data	Conditional[4]	96
	$EA	ECU Configuration	Conditional[2]	97
	$EB	Diagnostic Protocol Information	Conditional[5,6]	97
	$EC-$EF	**Reserved**	N/A	
	$F0-$F9	Dynamically Defined Local Identifiers	Optional	97
	$FA-$FE	System Supplier Specific	N/A	105
	$FF	Reserved	N/A	

Table 3.8.2-1 Read Data By Local Identifier Request Message Format

Condition[1]: Development data shall be supported by all ECUs on the CAN-Bus
Condition[2]: Shall be supported by ECUs that are used at DCA.
Condition[3]: Flash Info shall be used during development if the flash loader software module (version ≥ 1.0) is used for flash reprogramming. (See Reference "*M"* in "Appendix A – *References*")
Condition[4]: (DCS only) System diagnostic parameter data shall be provided by all ECUs that system diagnosis communication module SD-COM.
Condition[5]: Refer to the ECU Flash Reprogramming Specification for condition under which this LID must be supported. (See Reference "L" in "Appendix A – *References*")
Condition[6]: Shall be supported by ECUs if required by the diagnostic performance standards (See Reference "H" and "I" in "Appendix A – *References*")

3.8.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$61**	**Read Data By Local Identifier Positive Response Service ID**	**Mandatory**	
1	**$XX**	**Record Local Identifiers** The **Record Local Identifier** parameter must be identical to the **Record Local Identifier** parameter sent in the request message. Refer to Table 3.8.2-1.	**Mandatory**	**85**
2	**$XX**	**Record Value #1**	**Mandatory**	**98**
:	**:**	**:**		
N	**$XX**	**Record Value #m**	Conditional	**98**

Table 3.8.3-1 Read Data By Local Identifier Positive Response Message Format

Condition: Record Value #m shall be reported if data length exceeds 1 byte.

3.8.4 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Mandatory**	
1	**$21**	**Read Data By Local Identifier Request Service ID**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.8.4-1 Read Data By Local Identifier Negative Response Message

3.8.5 IMPLEMENTATION EXAMPLE OF **READ DATA BY LOCAL IDENTIFIER** (RECORD LOCAL IDENTIFIER = ENGINE DATA ($0A))

Hex	Diagnostic Tool Request Message
$21	**Read Data By Local Identifier Request**
$0A	Record Local Identifier = Record #10

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$61	**Read Data By Local Identifier Positive Response**	**$7F**	**Negative Response Service Identifier**
$0A	Record Local Identifier = Engine Data ($0A)	$21	Read Data By Local Identifier Request
$8C	Record Value #1 (Battery Positive Voltage)	$XX	Negative Response Code
$A6	Record Value #2 (Engine Coolant Temperature)		
$66	Record Value #3 (Throttle Position Sensor)		
$A0	Record Value #4 (Oxygen Sensor)		

Table 3.8.5-1 Implementation Example of Read Local Identifier

3.8.6 IMPLEMENTATION EXAMPLE OF **READ DATA BY LOCAL IDENTIFIER** (RECORD LOCAL IDENTIFIER = ECU SERIAL NUMBER ($E1))

Hex	Diagnostic Tool Request Message
$21	**Read Data By Local Identifier Request**
$E1	Record Local Identifier = ECU Serial Number

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$61	**Read Data By Local Identifier Positive Response**	**$7F**	**Negative Response Service Identifier**
$E1	Record Local Identifier = ECU Serial Number	$21	Read Data By Local Identifier Request
$12	ECU Serial Number (High Byte)	$XX	Negative Response Code
$34	ECU Serial Number		
$56	ECU Serial Number		
$78	ECU Serial Number (Low Byte)		

Table 3.8.6-1 Implementation Example Read ECU Serial Number

3.9 SERVICE ID $22 – *READ DATA BY IDENTIFIER*

3.9.1 MESSAGE DESCRIPTION

PURPOSE

The service, **Read Data By Identifier ($22),** requests blocks of data from the ECU. The blocks of data read may include sensor information, actuator status, and may be of varying length. The data is written to the same **Identifier** by using the **Write Data By Identifier ($2E)** service.

3.9.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$22**	**Read Data By Identifier Request Service ID**	**Mandatory**	
1	**$XX**	**Identifier (MSB)**	**Mandatory**	**85**
2	**$XX**	**Identifier (LSB)**	**Mandatory**	**85**
	$0000 - $00FF	Reserved	N/A	
	$0100 - $EFFF	Identifiers	Optional	85
	$F000 - $F0FF	Reserved	**Mandatory**	
	$F100 - $F19F	Identifiers	Optional	**85**
	$F1F0 - $F1FF	Supplier Specific	Optional	105
	$F200 - $FCFF	Reserved	**Mandatory**	
	$FD00 - $FEFF	Supplier Specific	Optional	105
	$FF00 - $FFFF	Reserved	**Mandatory**	

Table 3.9.2-1 Read Data By Identifier Request Message Format

3.9.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$62**	**Read Data By Identifier Positive Response Service ID**	**Mandatory**	
1	**$XX**	**Identifier (MSB)**	**Mandatory**	**85**
2	**$XX**	**Identifier (LSB)**	**Mandatory**	**85**
		The **Identifier** must be identical to the **Identifier** sent in the request message.		
3	**$XX**	**Record Value #1**	**Mandatory**	**98**
:	**:**	**:**		
N	**$XX**	**Record Value #m**	Conditional	**98**

Table 3.9.3-1 Read Data By Identifier Positive Response Message Format

Condition: Record Value #m shall be reported if data length exceeds 1 byte.

3.9.4 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Mandatory**	
1	**$22**	**Read Data By Identifier Request Service ID**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to KWP2000 Requirements Specification for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.9.4-1 Read Data By Identifier Negative Response Message

3.9.5 IMPLEMENTATION EXAMPLE OF **READ DATA BY IDENTIFIER** (IDENTIFIER = ENGINE DATA)

Hex	Diagnostic Tool Request Message
$22	**Read Data By Identifier Request**
$0A	Identifier (MSB) = Engine Data
$00	Identifier (LSB) = Engine Data

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$62	**Read Data By Identifier Positive Response**	**$7F**	**Negative Response Service Identifier**
$0A	Identifier (MSB) = Engine Data	$22	Read Data By Identifier Request
$00	Identifier (LSB) = Engine Data	$XX	Negative Response Code
$8C	Record Value #1 (Battery Positive Voltage)		
$A6	Record Value #2 (Engine Coolant Temperature)		
$66	Record Value #3 (Throttle Position Sensor)		
$A0	Record Value #4 (Oxygen Sensor)		

Table 3.9.5-1 Implementation Example of Read Data By Identifier

3.10 SERVICE ID $23 – *READ MEMORY BY ADDRESS*

3.10.1 MESSAGE DESCRIPTION

PURPOSE

The service, **Read Memory By Address ($23),** requests data stored in a 3 byte memory address from the ECU. The location of the data and the size of the data block is identified by **Memory Address** and **Memory Size**.

REQUIREMENTS

1) The **Read Memory By Address ($23)** service shall only be used during the period of ECU development. The **Read Data By Local ID ($21)** service shall be used in production and service.
2) Any ECU that must have access to a **Memory Address** (e.g. for debug-information) shall implement appropriate security mechanisms.

3.10.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$23**	**Read Memory By Address Request Service ID**	**Mandatory**	
1	**$XX**	**Memory Address (High Byte)**	**Mandatory**	**98**
2	**$XX**	**Memory Address (Middle Byte)**	**Mandatory**	**98**
3	**$XX**	**Memory Address (Low Byte)**	**Mandatory**	**98**
4	**$XX**	**Memory Size**	**Mandatory**	**98**

Table 3.10.2-1 Read Memory By Address Request Message Format

3.10.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$63**	**Read Memory By Address Positive Response Service ID**	**Mandatory**	
1	**$XX**	**Record Value #1**	**Mandatory**	**98**
:	**:**	**:**		
n	**$XX**	**Record Value #m**	Conditional	**98**

Table 3.10.3-1 Read Memory By Address Positive Response Message Format

Condition: Record Value #m shall be reported if Record Value Memory Size exceeds 1 byte.

3.10.4 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Mandatory**	
1	**$23**	**Read Data By Common Identifier Request Service ID**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.10.4-1 Read Memory By Address Negative Response Message

3.10.5 IMPLEMENTATION EXAMPLE OF **READ MEMORY BY ADDRESS** (MEMORY ADDRESS = $204813)

In the following example, the diagnostic tool reads 3 data bytes from the ECU's RAM cells starting at memory address $204813.

Hex	Diagnostic Tool Request Message
$23	**Read Memory Address Request**
$20	Memory Address (High Byte)
$48	Memory Address (Middle Byte)
$13	Memory Address (Low Byte)
$03	Memory Size

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$63	**Read Memory Address Positive Response**	**$7F**	**Negative Response Service Identifier**
$00	External RAM Cell #1	$23	Read Memory Address Request
$46	External RAM Cell #2	$XX	Negative Response Code
$FB	External RAM Cell #3		

Table 3.10.5-1 Implementation Example of Read Memory Address

3.11 SERVICE ID $27 – *SECURITY ACCESS*

3.11.1 MESSAGE DESCRIPTION

PURPOSE:

The service, **Security Access ($27)**, is intended to be used to implement data link security measures defined in SAE J2186 / ISO 15031-7 (E/E Data Link Security). Proper "unlocking" of the ECU is a prerequisite to the diagnostic tool's ability to perform some of the more critical functions such as reading specific memory locations within the ECU, downloading information to specific locations, or downloading routines for execution by the controller. This permits the ECU specific software to protect itself from unauthorized intrusion. Note for DCA: Refer to the DCA Security Application Guidelines (refer to reference P) for additional information.

REQUIREMENTS:

1. ECU's which provide security access shall support the negative response, "Security Access Denied / Security Access Requested ($33)", if a secured service is requested while the ECU is locked.
2. An ECU shall power up in a "locked" state.
3. Upon returning to a **Normal/Default Session ($81)**, the ECU shall return to a "locked" state.
4. The Seed and Key shall each be a minimum of 2 bytes in length. Selection of the minimum number of bytes will result in a minimum security level. Use of 4 or more bytes is recommended.
5. The following sequence of services shall be implemented in order to support/perform secured functions.
    1. **Start Diagnostic Session ($10)**
    2. **Security Access ($27)**
6. The following procedure shall be used for "unlocking" an ECU:
    1. The diagnostic tool shall send the **Security Access (Request Seed) Request ($27)** service requesting the **Seed** parameter.
    2. The ECU shall respond by sending the **Seed** parameter using the **Security Access (Request Seed) Positive Response ($67)** service.
    3. The diagnostic tool shall respond by returning the **Key** back to the ECU using the **Security Access (Send Key) Request ($27)** service.
    4. The ECU shall compare this **Key** to one internally calculated. If the two numbers match, then the ECU shall enable ("unlock") the diagnostic tool's access to specific KWP 2000 services. This shall be indicated to the diagnostic tool via the **Security Access (Send Key) Positive Response ($67)** service.
    5. After 2 unsuccessful attempts of **Security Access** procedures by the diagnostic tool, the ECU shall insert a 10 second time delay before allowing further attempts.
    6. A 10 second time delay shall be required before the ECU responds to a **Security Access (Request Seed) Request ($27)** service from the diagnostic tool after ECU power-on if there have been 2 unsuccessful attempts of **Security Access**.
7. An ECU that supports security, but is already unlocked when a **Security Access (Request Seed) Request ($27)** service is received, shall respond with a **Security Access (Request Seed) Positive Response ($67)** service with a **Seed** of all "zero's". A diagnostic tool shall use this method to determine if a ECU is locked by checking for a non-zero seed.
8. An odd number for the subfunction **Access Mode** indicates **Request Seed**, and the following even number is the corresponding subfunction **Send Key.**
9. The security system shall not prevent normal diagnostic or vehicle communications. (i.e. Any service supported by the **Normal/Default Session ($81)** parameter.)
10. The only access to the ECU permitted while in a "locked" mode is through the ECU specific software.
11. In case the diagnostic tool does not recognize the **Security Access (Request Seed) Positive Response ($67)** service of the ECU, the ECU shall respond to a further **Security Access (Request Seed) Request ($27)** services with the **Security Access (Request Seed) Positive Response ($67)** services. It shall not respond with **Request Sequence Error ($22)** or **Invalid Key ($35)** negative response codes**.**
12. $FF..FF (all F's) shall not be used for the **Seed** parameter because this may occur if the ECU's memory has been erased.

3.11.2 REQUEST MESSAGE FORMAT (REQUEST SEED)

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$27**	**Security Access (Request Seed) Request Service ID**	**Mandatory**	
1	**$XX**	**Access Mode**	**Mandatory**	**98**
	$00	Reserved	N/A	
	$01, $03, $05, $07-$7F	Request Seed	Mandatory	98
	$81-$FA	Reserved	N/A	
	$FB-$FE	System Supplier Specific	N/A	105
	$FF	Reserved	N/A	

Table 3.11.2-1 Security Access (Request Seed) Request Message Format

3.11.3 POSITIVE RESPONSE MESSAGE FORMAT (REQUEST SEED)

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$67**	**Security Access (Request Seed) Positive Response Service ID**	**Mandatory**	
1	**$XX**	**Access Mode** The **Access Mode** parameter must be identical to the **Access Mode** parameter sent in the request message. Refer to Table 3.11.2-1	**Mandatory**	**98**
2	**$XX**	**Seed (High Byte)**	**Mandatory**	**99**
:	**:**	**:**		
n	**$XX**	**Seed (Low Byte)**	**Conditional**	**99**

Table 3.11.3-1 Security Access (Request Seed) Positive Response Message Format

Condition: Seed (Low Byte) shall be reported if Seed length exceeds 1 byte.

3.11.4 NEGATIVE RESPONSE MESSAGE (REQUEST SEED)

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Mandatory**	
1	**$27**	**Security Access Request Service ID**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.11.4-1 Security Access (Request Seed) Negative Response Message

3.11.5 REQUEST MESSAGE FORMAT (SEND KEY)

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$27**	**Security Access (Send Key) Request Service ID**	**Mandatory**	
1	**$XX**	**Access Mode**	**Mandatory**	**98**
	$00	Reserved	N/A	
	$02, $04,$06, $08-$80	Send Key	Mandatory	98
	$81-$FA	Reserved	N/A	
	$FB-$FE	System Supplier Specific	N/A	105
	$FF	Reserved	N/A	
2	**$XX**	**Key (High Byte)**	**Mandatory**	**98**
:	**:**	**:**		
n	**$XX**	**Key (Low Byte)**	**Conditional**	**98**

Table 3.11.5-1 Security Access (Send Key) Request Message Format

Condition: Key (Low Byte) shall be reported if Key length exceeds 1 byte.

3.11.6 POSITIVE RESPONSE MESSAGE FORMAT (SEND KEY)

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$67**	**Security Access (Send Key) Positive Response Service ID**	**Mandatory**	
1	**$XX**	**Access Mode** The **Access Mode** parameter must be identical to the **Access Mode** parameter sent in the request message. Refer to Table 3.11.5-1.	**Mandatory**	**98**
2	**$34**	**Security Access Status = [Security Access Allowed]**	**Mandatory**	**99**

Table 3.11.6-1 Security Access (Send Key) Positive Response Message Format

3.11.7 NEGATIVE RESPONSE MESSAGE (SEND KEY)

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Mandatory**	
1	**$27**	**Security Access Request Service ID**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.11.7-1 Security Access (Send Key) Negative Response Message

3.11.8 IMPLEMENTATION EXAMPLE OF **SECURITY ACCESS**(ACCESS MODE = REQUEST SEED (ECU LOCKED))

This section specifies the conditions to be fulfilled to successfully unlock the ECU if it is in a "locked" state:
Request Seed = $01; **Send Key** = $02; **Seed** = $36218575; **Key** = $C93E2F8B.
e.g. Secure Algorithm f(x); f($36218575)=$C93E2F8B

Hex	Diagnostic Tool Request Message
$27	**Security Access Request**
$01	Access Mode = Request Seed

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$67	**Security Access Positive Response**	**$7F**	**Negative Response Service Identifier**
$01	Access Mode = Request Seed	$27	Security Access Request
$36	Seed (High Byte)	$XX	Negative Response Code
$21	Seed		
$85	Seed		
$75	Seed (Low Byte)		

Hex	Diagnostic Tool Request Message
$27	**Security Access Request**
$02	Access Mode = Send Key
$C9	Key (High Byte)
$3E	Key
$2F	Key
$8B	Key (Low Byte)

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$67	**Security Access Positive Response**	**$7F**	**Negative Response Service Identifier**
$02	Access Mode = Send Key	$27	Security Access Request
$34	Security Access Allowed	$XX	Negative Response Code

Table 3.11.8-1 Implementation Example of Security Access (Request Seed) / ECU Locked

3.11.9 IMPLEMENTATION EXAMPLE OF **SECURITY ACCESS**(ACCESS MODE = REQUEST SEED (ECU UNLOCKED))

The following example shows the message flow if ECU is in an "unlocked" state (seed = $00000000).

Hex	Diagnostic Tool Request Message
$27	**Security Access Request**
$01	Access Mode = Send Key

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$67	**Security Access Positive Response**	**$7F**	**Negative Response Service Identifier**
$01	Access Mode = Request Seed	$27	Security Access Request
$00	Seed (High Byte)	$XX	Negative Response Code
$00	Seed		
$00	Seed		
$00	Seed (Low Byte)		

Table 3.11.9-1 Implementation Example Security Access (Request Seed) / ECU Unlocked

3.12 SERVICE ID $28 – *DISABLE NORMAL MESSAGE TRANSMISSION*

3.12.1 MESSAGE DESCRIPTION

PURPOSE:
The service, **Disable Normal Message Transmission ($28)**, is used to "switch off" the transmission path of non-diagnostic and non-network management messages.

REQUIREMENTS:
1. Conditions for controlling transmission are as follows:
   i. Switch off transmission path:
      - through diagnostic service **Disable Normal Message Transmission ($28)**
   ii. Switch transmission path on again:
      - through diagnostic service **Enable Normal Message Transmission ($29)**
      - after loss of ECU supply voltage (normal ECU powerdown event)
      - when switching from the Extended Diagnostic Session to any other Diagnostic Session, especially the Default Session.

2. After an ECU returns the positive response to the Disable Normal Message Transmission Request, the functionality of Network Management shall remain unaffected.

3.12.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$28**	**Disable Normal Message Transmission Request Service ID**	**Mandatory**	
1	**$XX**	**Response Required**	**Mandatory**	**99**
	$01	Response Required	Mandatory	99
	$02	No Response Required	Mandatory	99

Table 3.12.2-1 Disable Normal Message Transmission Request Message Format

3.12.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$68**	**Disable Normal Message Transmission Positive Response Service ID**	**Conditional**	

Table 3.12.3-1 Disable Normal Message Transmission Positive Response Message Format

Condition: An ECU shall not return a **Disable Normal Message Positive Response ($68)** if the **No Response Required ($02)** parameter was used by the diagnostic test tool in the request.

3.12.4 NEGATIVE RESPONSE MESSAGE

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Conditional**	
1	**$28**	**Disable Normal Message Transmission Request Service ID**	**Conditional**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Conditional**	

Table 3.12.4-1 Disable Normal Message Transmission Negative Response Message

Condition: An ECU shall not return a **Negative Response ($7F)** if the **No Response Required ($02)** parameter was used by the diagnostic test tool in the request.

3.12.5 IMPLEMENTATION EXAMPLE OF **DISABLE NORMAL MESSAGE TRANSMISSION** (RESPONSE REQUIRED = NO)

The following example shows the message flow if the ECU is in normal operation mode.

Hex	Diagnostic Tool Request Message
$28	**Disable Normal Message Transmission Request Service ID**
$02	No Response Required

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
	No Response		No Response

Table 3.12.5-1 Implementation Example of Disable Normal Message Transmission

3.13 SERVICE ID $29 – *ENABLE NORMAL MESSAGE TRANSMISSION*

3.13.1 MESSAGE DESCRIPTION

PURPOSE:
The service, **Enable Normal Message Transmission ($29)**, is used to "switch on" the transmission path for all messages. This service is used in conjunction with the **Disable Normal Message Transmission ($28)** service.

3.13.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$29**	**Enable Normal Message Transmission Request Service ID**	**Mandatory**	
1	**$XX**	**Response Required**	**Mandatory**	**99**
	$01	Response Required	Mandatory	99
	$02	No Response Required	Mandatory	99

Table 3.13.2-1 Enable Normal Message Transmission Request Message Format

3.13.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$69**	**Enable Normal Message Transmission Positive Response Service ID**	**Conditional**	

Table 3.13.3-1 Enable Normal Message Transmission Positive Response Message Format

Condition: An ECU shall not return a **Disable Normal Message Positive Response ($69)** if the **No Response Required ($02)** parameter was used by the diagnostic test tool in the request.

3.13.4 NEGATIVE RESPONSE MESSAGE

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Conditional[1]**	
1	**$29**	**Enable Normal Message Transmission Request Service ID**	**Conditional[1]**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Conditional[1]**	

Table 3.13.4-1 Enable Normal Message Transmission Negative Response Message

Condition[1]: An ECU shall not return a **Negative Response ($7F)** if the **No Response Required ($02)** parameter was used by the diagnostic test tool in the request.

3.13.5 IMPLEMENTATION EXAMPLE OF **ENABLE NORMAL MESSAGE TRANSMISSION** (RESPONSE REQUIRED = YES)

The following example shows the message flow if the ECU is in disable normal message transmission mode.

Hex	Diagnostic Tool Request Message
$29	**Enable Normal Message Transmission Request Service ID**
$01	Response Required

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$69	**Enable Normal Message Transmission Positive Response**	**$7F**	**Negative Response Service Identifier**
		$29	Enable Normal Message Transmission Request
		$XX	Negative Response Code

Table 3.13.5-1 Implementation Example of Enable Normal Message Transmission

3.14 SERVICE ID $2C – *DYNAMICALLY DEFINE LOCAL IDENTIFIER*

3.14.1 MESSAGE DESCRIPTION

PURPOSE:
The service, **Dynamically Define Local Identifier ($2C)**, is used to dynamically create **Record Local Identifiers**. If no previously defined **Record Local Identifier** contains all the desired data, pieces of various data sets may be selected and put together.

REQUIREMENTS:
1. This service shall use local identifiers within the range of $F0 to $F9. (See Table 3.8.2-1)
2. **Read Data By Local Identifier ($21)** shall be used to read the local identifier dynamically defined with this service.

3.14.2 REQUEST MESSAGE FORMATS

The structure of the request messages will vary based upon the **Definition Mode** parameter. The following tables show the proper request message structure for dynamically defining a Local Identifier from previously defined **Record Local Identifiers**.

REQUEST MESSAGE FOR DEFINITION MODE = DEFINE BY LOCAL IDENTIFIER ($01)

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$2C**	**Dynamically Define Local Identifier Request Service Id**	**Mandatory**	
1	**$XX**	**Dynamically Defined Local Identifier**	**Mandatory**	**97**
2	**$XX**	**Definition Mode #1**	**Mandatory**	**97**
	$01	Define By Local Identifier	Optional	98
3	**$XX**	**Position In Dynamically Defined Local Identifier #1**	**Mandatory**	**98**
4	**$XX**	**Memory Size #1**	**Mandatory**	**98**
5	**$XX**	**Local Identifier $XX**	**Mandatory**	**98**
6	**$XX**	**Position In Record Local Identifier $XX**	**Mandatory**	**98**
7	**$XX**	**Definition Mode #2**	**Conditional**	**97**
	$01	Define By Local Identifier	**Conditional**	98
8	**$XX**	**Position In Dynamically Defined Local Identifier #2**	**Conditional**	**98**
9	**$XX**	**Memory Size #2**	**Conditional**	**98**
10	**$XX**	**Local Identifier $XX**	**Conditional**	**98**
11	**$XX**	**Position In Record Local Identifier $XX**	**Conditional**	**98**
:	**:**	**:**	**:**	
n-4	**$XX**	**Definition Mode #m**	**Conditional**	**97**
	$01	Define By Local Identifier	**Conditional**	98
n-3	**$XX**	**Position In Dynamically Defined Local Identifier #m**	**Conditional**	**98**
n-2	**$XX**	**Memory Size #m**	**Conditional**	**98**
n-1	**$XX**	**Local Identifier $XX**	**Conditional**	**98**
n	**$XX**	**Position In Record Local Identifier $XX**	**Conditional**	**98**

Table 3.14.2-1 Dynamically Define Local Identifier (Definition Mode=$01) Request Message Format

Condition: Bytes 7 to n are required for any ECU which will be required to support m local identifiers in the Dynamically Defined Local Identifier.

REQUEST MESSAGE FOR DEFINITION MODE = DEFINE BY MEMORY ADDRESS ($02)

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$2C**	**Dynamically Define Local Identifier Request Service Id**	**Mandatory**	
1	**$XX**	**Dynamically Defined Local Identifier**	**Mandatory**	**97**
2	**$XX**	**Definition Mode #2**	**Mandatory**	**98**
	$02	Define By Memory Address	Optional	98
3	**$XX**	**Position In Dynamically Defined Local Identifier #1**	**Mandatory**	**98**
4	**$XX**	**Memory Size #1**	**Mandatory**	**98**
5	**$XX**	**Memory Address #1 (LSB)**	**Mandatory**	**98**
6	**$XX**	**Memory Address #1**	**Mandatory**	**98**
7	**$XX**	**Memory Address #1 (MSB)**	**Mandatory**	**98**
9	**$XX**	**Definition Mode #2**	**Conditional**	**98**
	$02	Define By Memory Address	**Conditional**	98
10	**$XX**	**Position In Dynamically Defined Local Identifier #1**	**Conditional**	**98**
11	**$XX**	**Memory Size #1**	**Conditional**	**98**
	$XX	**Memory Address #1 (LSB)**	**Conditional**	**98**
	$XX	**Memory Address #1**	**Conditional**	**98**
12	**$XX**	**Memory Address #1 (MSB)**	**Conditional**	**98**
:	**:**	**:**	**:**	
n-4	**$XX**	**Definition Mode #2**	**Conditional**	**98**
	$02	Define By Memory Address	**Conditional**	98
n-3	**$XX**	**Position In Dynamically Defined Local Identifier #1**	**Conditional**	**98**
n-2	**$XX**	**Memory Size #m**	**Conditional**	**98**
n-1	**$XX**	**Memory Address #m (LSB)**	**Conditional**	**98**
	$XX	**Memory Address #m**	**Conditional**	**98**
	$XX	**Memory Address #m (MSB)**	**Conditional**	**98**

Table 3.14.2-2 Dynamically Define Local Identifier (Definition Mode=$03) Request Message Format

Condition: Bytes 7 to n are required for any ECU which will be required to support m local identifiers in the Dynamically Defined Local Identifier.

Note: The Definition Mode $02 (Define By Memory Address) is intended for development only and must be removed for production.)

REQUEST MESSAGE FOR DEFINITION MODE = DEFINE BY IDENTIFIER ($03)

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$2C**	**Dynamically Define Local Identifier Request Service Id**	**Mandatory**	
1	**$XX**	**Dynamically Defined Local Identifier**	**Mandatory**	**97**
2	**$XX**	**Definition Mode #3**	**Mandatory**	**98**
	$03	Define By Identifier	**Conditional**	98
3	**$XX**	**Position In Dynamically Defined Local Identifier# 1**	**Mandatory**	**98**
4	**$XX**	**Memory Size #1**	**Mandatory**	**98**
5	**$XXXX**	**Identifier $XXXX**	**Mandatory**	**85**
7	**$XX**	**Position In Identifier $XXXX**	**Mandatory**	**98**
8	**$XX**	**Definition Mode #3**	**Conditional**	**98**
	$03	Define By Identifier	**Conditional**	98
8	**$XX**	**Position In Dynamically Defined Local Identifier# 1**	**Conditional**	**98**
9	**$XX**	**Memory Size #2**	**Conditional**	**98**
10	**$XXXX**	**Identifier $XXXX**	**Conditional**	**85**
11	**$XX**	**Position In Identifier $XXXX**	**Conditional**	**98**
:	**:**	**:**	**:**	
n-4	**$XX**	**Definition Mode #3**	**Conditional**	**98**
	$03	Define By Identifier	**Conditional**	98
n-3	**$XX**	**Position In Dynamically Defined Local Identifier# 1**	**Conditional**	**98**
n-2	**$XX**	**Memory Size #m**	**Conditional**	**98**
n-1	**$XXXX**	**Identifier $XXXX**	**Conditional**	**85**
n	**$XX**	**Position In Identifier $XXXX**	**Conditional**	**98**

Table 3.14.2-3 Dynamically Define Local Identifier (Definition Mode=$03) Request Message Format

Condition: Bytes 7 to n are required for any ECU which will be required to support m local identifiers in the Dynamically Defined Local Identifier.

REQUEST MESSAGE FOR DEFINITION MODE = DEFINE BY LOCAL IDENTIFIER ($04)

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$2C**	**Dynamically Define Local Identifier Request Service Id**	**Mandatory**	
1	**$XX**	**Dynamically Defined Local Identifier**	**Mandatory**	**97**
2	**$XX**	**Definition Mode #1**	**Mandatory**	**97**
3	$04	Clear Dynamically Defined Local Identifier	Optional	98

Table 3.14.2-4 Dynamically Define Local Identifier (Definition Mode=$04) Request Message Format

3.14.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$6C**	**Dynamically Define Local Identifier Positive Response Service Id**	**Mandatory**	
1	**$XX**	**Dynamically Defined Local Identifier**	**Mandatory**	**97**

Table 3.14.3-1 Dynamically Define Local Identifier Positive Response Message Format

3.14.4 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Mandatory**	
1	**$2C**	**Dynamically Define Local Identifier Request Service ID**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.14.4-1 Dynamically Define Local Identifier Negative Response Message

3.14.5 IMPLEMENTATION EXAMPLE OF **DYNAMICALLY DEFINE LOCAL IDENTIFIER** (DEFINITION MODE = READ LOCAL IDENTIFIER)

The following example shows the process for using a **Dynamically Define Local Identifier** and **Read Data By Local Identifier** services.

1. **Dynamically Define Local ID Request ($2C)** [from Diagnostic Tester]
2. **Dynamically Define Local ID Positive Response ($6C)** [from ECU]
3. **Read Data By Local ID Request ($21)** [from Diagnostic Tester]
4. **Read Data By Local ID Positive Response ($61)** [from ECU]

(Refer to see Table 3.14.5-3)

For this example, three local identifiers ($A0, $A1, $A2) are stored in the ECU prior to dynamically defining the new local ID. These local identifiers are defined as follows:

	Position 1	Position 2	Position 3	Position 4
Local Identifier=$A0	Engine Coolant Temp=$38	Oil Temp(MSB)=$40	Oil Temp(LSB)=$F4	Oil Pressure=$41
Local Identifier=$A1	Engine Speed=$5E	Throttle Position=$2F	Vehicle Speed=$0E	
Local Identifier=$A2	Current Gear=$03	Requested Gear=$01	Torque(MSB)=$0E	Torque(LSB)=$01

Table 3.14.5-1 Data for the Dynamically Define Local Identifier(Definition Mode = Define By Local ID) Request ($2C) message

The **Dynamically Define Local Identifier** service is used to create a new dynamically defined local identifier $F0. The content of this new dynamically defined local identifier ($F0) is Oil Temperature, Engine Speed, and Torque as shown in Table 3.14.5-2.

	Position 1	Position 2	Position 3	Position 4	Position 5
Dynamically Defined Local Identifier=$F0	Oil Temp(MSB)=$40	Oil Temp(LSB)=$F4	Engine Speed=$5E	Torque(MSB)=$0E	Torque(LSB)=$01

Table 3.14.5-2 Dynamically Defined Local Identifier ($F0)

Hex	Diagnostic Tool Request Message
$2C	Dynamically Defined Local Identifier Request
$F0	Dynamically Defined Local Identifier #1
$01	Definition Mode = Define By Local Identifier #1
$01	Position In Dynamically Defined Local Identifier #1
$02	Memory Size #1
$A0	Record Local Identifier (Data stream which includes Oil Temperature) #1
$02	Position In Record Local Identifier #1
$01	Definition Mode = Define By Local Identifier #2
$03	Position In Dynamically Defined Local Identifier #2
$01	Memory Size #2
$A1	Record Local Identifier (Data stream which includes Engine Speed) #2
$01	Position In Record Local Identifier #2
$01	Definition Mode = Define By Local Identifier #3
$04	Position In Dynamically Defined Local Identifier #3
$02	Memory Size #3
$A2	Record Local Identifier (Data stream which includes Torque) #3
$03	Position In Record Local Identifier #3

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$6C	Dynamically Defined Local Identifier Positive Response	$7F	Negative Response Service Identifier
$F0	Dynamically Defined Local Identifier	$2C	Dynamically Defined Local Identifier Request
		$XX	Negative Response Code

Table 3.14.5-3 Implementation Example of Dynamically Defined Local Identifier ($F0) - Part A

After completion of the **Dynamically Defined Record Local Identifier**, the new record shall be read by the service **Read Data By Local Identifier Request ($21).** The **Record Local Identifier** is set to '$F0' to read the new dynamically defined local identifier defined in Table 3.14.2-4.

Hex	Diagnostic Tool Request Message
$21	Read Data By Local Identifier Request
$F0	Dynamically Defined Local Identifier

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$61	Read Data By Local Identifier Positive Response	$7F	Negative Response Service Identifier
$F0	Dynamically Defined Local Identifier	$21	Read Data By Local Identifier Request
$40	Record Value #1 (Oil Temperature(MSB))	$XX	Negative Response Code
$F4	Record Value #1 (Oil Temperature(LSB))		
$5E	Record Value #2 (Engine Speed)		
$0E	Record Value #3 (Torque (MSB))		
$01	Record Value #3 (Torque(LSB))		

Table 3.14.5-4 Implementation Example of Dynamically Defined Local Identifier ($F0) - Part B

3.14.6 IMPLEMENTATION EXAMPLE OF **DYNAMICALLY DEFINE LOCAL IDENTIFIER** (DEFINITION MODE = CLEAR DYNAMICALLY DEFINED LOCAL IDENTIFIER)

This example deletes the dynamically defined **Record Local Identifier** “F0“ defined in example #1.

Hex	Diagnostic Tool Request Message
$2C	**Dynamically Defined Local Identifier Request**
$F0	Dynamically Defined Local Identifier
$04	Definition Mode = Clear Dynamically Defined Local Identifier

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$6C	**Dynamically Defined Local Identifier Positive Response**	**$7F**	**Negative Response Service Identifier**
$F0	Dynamically Defined Local Identifier	$2C	Dynamically Defined Local Identifier Request
		$XX	Negative Response Code

Table 3.14.6-1 Implementation Example of Clear Dynamically Defined Local Identifier

3.15 SERVICE ID $2E – *WRITE DATA BY IDENTIFIER*

3.15.1 MESSAGE DESCRIPTION

PURPOSE

The service, **Write Data By Identifier ($2E),** is used by the diagnostic tool to write **Record Values** (data values) to an ECU. The **Identifier** data is read by using the **Record Identifier ($22)**.

3.15.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$2E**	**Write Data By Identifier Request Service Id**	**Mandatory**	
1	**$XX**	**Identifier (MSB)**	**Mandatory**	**85**
2	**$XX**	**Identifier (LSB)**	**Mandatory**	**85**
	$0000 - $00FF	Reserved	N/A	
	$0100 - $EFFF	Identifiers	Optional	85
	$F000 - $F0FF	Reserved	**Mandatory**	
	$F100 - $F19F	Identifiers	Optional	**85**
	$F1F0 - $F1FF	Supplier Specific	Optional	105
	$F200 - $FCFF	Reserved	**Mandatory**	
	$FD00 - $FEFF	Supplier Specific	Optional	105
	$FF00 - $FFFF	Reserved	**Mandatory**	
3	**$XX**	**Record Value #1**	**Mandatory**	**98**
:	**:**			
n	**$XX**	**Record Value #m**	**Conditional**	**98**

Table 3.15.2-1 Write Data By Identifier Request Message Format

Condition: Record Value #m shall be reported if data length exceeds 1 byte.

3.15.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$6E**	**Write Data By Identifier Positive Response Service Id**	**Mandatory**	
1	**$XX**	**Identifier (MSB)**	**Mandatory**	**85**
2	**$XX**	**Identifier (LSB)**	**Mandatory**	**85**
		The **Identifier** must be identical to the **Identifier** sent in the request message.		

Table 3.15.3-1 Write Data By Identifier Positive Response Message Format

3.15.4 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Mandatory**	
1	**$2E**	**Write Data By Identifier Request Service ID**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to KWP2000 Requirements Specification for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.15.4-1 Write Data By Identifier Negative Response Message

3.15.5 IMPLEMENTATION EXAMPLE OF **WRITE DATA BY IDENTIFIER**

The diagnostic tool shall write the Odometer Mileage data to the ECU by using the **Write Data By Identifier** service. The ECU shall send a **Write Data By Identifier Positive Response ($6E)** message after it has successfully programmed the data.

Hex	Diagnostic Tool Request Message
$2E	**Write Data By Identifier Request**
$04	Identifier (MSB) = Odometer Mileage
$50	Identifier (LSB) = Odometer Mileage
$01	Odometer (MSB)
$86	Odometer
$A0	Odometer (LSB)

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$6E	**Write Data By Identifier Positive Response**	**$7F**	**Negative Response Service Identifier**
$04	Identifier (MSB) = Odometer Mileage	$2E	Write Data By Identifier Request
$50	Identifier (LSB) = Odometer Mileage	$XX	Negative Response Code

Table 3.15.5-1 Implementation Example of Write Data By Identifier

3.16 SERVICE ID $30 – INPUT OUTPUT CONTROL BY LOCAL IDENTIFIER

3.16.1 MESSAGE DESCRIPTION

PURPOSE

The service, **Input Output Control By Local Identifier ($30),** is used by the diagnostic tool to substitute a value for an input signal, internal ECU function and/or control an output (actuator) of an electronic system.

REQUIREMENTS

1. After an ECU returns a positive response to this service, the diagnostic tool shall assume control over the inputs/outputs specified by the Input Output Local Identifier.
2. After returning control to the ECU either by requesting **Return Control to ECU ($30 $XX $00)**, returning to a Normal/Default Diagnostic Session, ECU Reset, or internal disable conditions (e.g. to prevent damage to the ECU or for security reasons) the ECU switches back to normal operation. (i.e. The ECU re-assumes control over the I/O previously under control of the diagnostic tool.)
3. When using the **Short Term Adjustment ($07)** parameter, the **Control State** used in the diagnostic request shall use the same values defined for the **Control State** in the **Report Current State ($01)** diagnostic response.

3.16.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$30**	**Input Output Control By Local Identifier Request Service Id**	**Mandatory**	
1	**$XX**	**Input Output Control By Local Identifier**	**Mandatory**	**99**
	$00	Reserved	N/A	
	$01 - $0F	Reserved	N/A	
	$10-$F9	Input Output Local Identifier	Optional	99
	$FA-$FE	System Supplier Specific	N/A	
	$FF	Input Output Local Identifier	Optional	99
2	**$XX**	**Input Output Control Parameter**	**Mandatory**	**99**
	$00	Return Control to ECU	Mandatory	99
	$01	Report Current State	Mandatory	99
	$02 -$03	Reserved	N/A	
	$04	Reset to Default	Optional	99
	$05	Freeze Current State	Optional	99
	$06	Reserved	N/A	
	$07	Short Term Adjustment (Actuate)	Mandatory	99
	$08	Long Term Adjustment	Conditional	99
	$09-$FF	Reserved	N/A	
3	**$XX**	**Control State #1**	**Optional**	**99**
:	**:**	**:**		
n	**$XX**	**Control State #m**	**Optional**	**99**

Table 3.16.2-1 Input Output Control By Local Identifier Request Message Format

Condition: This Input Output Control Parameter shall be used if the values applied by Short Term Adjustment shall be written permanently to the ECU. This parameter shall only be supported by DCS ECU's. Note: Data that is only permanently written to the ECU shall be applied with the service **Write Data By Local Identifier ($3B)**.

3.16.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$70**	**Input Output Control By Local Identifier Positive Response Service Id**	**Mandatory**	
1	**$XX**	**Input Output Local Identifier** The **Input Output Local Identifier** parameter must be identical to the **Input Output Local Identifier** parameter sent in the request message. Refer to Table 3.16.2-1	**Mandatory**	**99**
2	**$XX**	**Input Output Control Parameter** The **Input Output Local Parameter** must be identical to the **Input Output Local Parameter** sent in the request message. Refer to Table 3.16.2-1	**Mandatory**	**99**
3 **:** **n**	**$XX** **:** **$XX**	**Control State #1** **Control State #m**	**Optional** **Optional**	**99** **99**

Table 3.16.3-1 Input Output Control By Local Identifier Positive Response Message Format

3.16.4 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Mandatory**	
1	**$30**	**Input Output Control By Local Identifier Request Service Id**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.16.4-1 Input Output Control By Local Identifier Negative Response Message

3.16.5 IMPLEMENTATION EXAMPLE OF **INPUT OUTPUT CONTROL BY LOCAL IDENTIFIER** (WARNING LAMP ACTUATION)

In this example, the **Input Output Local Control By Local Identifier** service will be used to actuate various warning lamps. Table 3.16.5-1 shows the Control State bytes for the Input Output Local Identifier $10.

Note: For DCA, I/O is typically not grouped together into a single local identifier. In other words, multiple inputs are not grouped into a single local identifier nor are multiple outputs. By grouping inputs or outputs together, control is taken away from multiple I/O ports at each request. When an ECU is requested to actuate a single I/O within a group of I/O, other I/O within the group cannot react "naturally". Refer to the "Diagnostic Performance Standard" for more details concerning I/O implementation. (See reference "H" in "Appendix A – *References*")

	Control State Byte 1	Control State Byte 2
IO Local Identifier=$10	Bit 0: MIL	Bit 0: Service 4WD
	Bit 1:ABS Lamp	Bit 1: Service Transfer Case
	Bit 2: Seat Belt Lamp	Bit 2: Exterior Lamp Out
	Bit 3: Highbeam Lamp	Bit 3: Park Indicator
	Bit 4: Cruise Control Lamp	Bit 4: Reverse Indicator
	Bit 5: Brake Lamp	Bit 5: Neutral Indicator
	Bit 6: Security Lamp	Bit 6: Drive Indicator
	Bit 7: Airbag	Bit 7: Third Indicator

Table 3.16.5-1 Example of Bit Encoding for Input Output Control Local Identifier $10

To begin, the current state of each indicator will be read using the **Report Current State Control Parameter**.

Hex	Diagnostic Tool Request Message
$30	**Input Output Control By Local Identifier Request**
$10	Input Output Local Identifier
$01	Input Output Control Parameter = Report Current State

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$70	**Input Output Control By Local Identifier Positive Response**	**$7F**	**Negative Response Service Identifier**
$10	Input Output Local Identifier	$30	Input Output Control By Local Identifier Request
$01	Input Output Control Parameter = Report Current State	$XX	Negative Response Code
$83	Control State Byte #1		
	Bit 0: MIL = 1		
	Bit 1: ABS Lamp = 1		
	Bit 2: Seat Belt Lamp = 0		
	Bit 3: Highbeam Lamp = 0		
	Bit 4: Cruise Control Lamp = 0		
	Bit 5: Brake Lamp = 0		
	Bit 6: Security Lamp = 0		
	Bit 7: Airbag = 1		
$52	Control State Byte #2		
	Bit 0: Service 4WD = 0		
	Bit 1: Service Transfer Case = 1		
	Bit 2: Exterior Lamp Out = 0		
	Bit 3: Park Indicator = 0		
	Bit 4: Reverse Indicator = 1		
	Bit 5: Neutral Indicator = 0		
	Bit 6: Drive Indicator = 1		
	Bit 7: Third Indicator = 0		

Table 3.16.5-2 Implementation Example of IO Control Using Report Current State Control Parameter

Next, all indicators will be switched off using the **Short Term Adjustment Control Parameter**.

Hex	Diagnostic Tool Request Message
$30	**Input Output Control By Local Identifier Request**
$10	Input Output Local Identifier
$07	Input Output Control Parameter = Short Term Adjustment
$00	Control State Byte #1
$00	Control State Byte #2

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$70	**Input Output Control By Local Identifier Positive Response**	**$7F**	**Negative Response Service Identifier**
$10	Input Output Local Identifier	$30	Input Output Control By Local Identifier Request
$07	Input Output Control Parameter = Short Term Adjustment	$XX	Negative Response Code
$00	Control State Byte #1		
	Bit 0: MIL = 0		
	Bit 1:ABS Lamp = 0		
	Bit 2: Seat Belt Lamp = 0		
	Bit 3: Highbeam Lamp = 0		
	Bit 4: Cruise Control Lamp = 0		
	Bit 5: Brake Lamp = 0		
	Bit 6: Security Lamp = 0		
	Bit 7: Airbag = 0		
$00	Control State Byte #2		
	Bit 0: Service 4WD = 0		
	Bit 1: Service Transfer Case = 0		
	Bit 2: Exterior Lamp Out = 0		
	Bit 3: Park Indicator = 0		
	Bit 4: Reverse Indicator = 0		
	Bit 5: Neutral Indicator = 0		
	Bit 6: Drive Indicator = 0		
	Bit 7: Third Indicator = 0		

Table 3.16.5-3 Implementation Example of IO Control Using Short Term Adjustment Control Parameter – Part A

Next, the following lamps will be actuated using Short Term Adjustment Control Parameter: MIL, ABS, Seat Belt, Service 4WD, Park Indicator, Third Indicator.

Hex	Diagnostic Tool Request Message
$30	**Input Output Control By Local Identifier Request**
$10	Input Output Local Identifier
$07	Input Output Control Parameter = Short Term Adjustment
$07	Control State Byte #1
$89	Control State Byte #2

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$70	**Input Output Control By Local Identifier Positive Response**	**$7F**	**Negative Response Service Identifier**
$10	Input Output Local Identifier	$30	Input Output Control By Local Identifier Request
$07	Input Output Control Parameter = Short Term Adjustment	$XX	Negative Response Code
$07	Control State Byte #1		
	Bit 0: MIL = 1		
	Bit 1: ABS Lamp = 1		
	Bit 2: Seat Belt Lamp = 1		
	Bit 3: Highbeam Lamp = 0		
	Bit 4: Cruise Control Lamp = 0		
	Bit 5: Brake Lamp = 0		
	Bit 6: Security Lamp = 0		
	Bit 7: Airbag = 0		
$89	Control State Byte #2		
	Bit 0: Service 4WD = 1		
	Bit 1: Service Transfer Case = 0		
	Bit 2: Exterior Lamp Out = 0		
	Bit 3: Park Indicator = 1		
	Bit 4: Reverse Indicator = 0		
	Bit 5: Neutral Indicator = 0		
	Bit 6: Drive Indicator = 0		
	Bit 7: Third Indicator = 1		

Table 3.16.5-4 Implementation Example of IO Control Using Short Term Adjustment Control Parameter – Part B

Finally, the diagnostic tool will return control to the ECU using the **Return Control to ECU** Control Parameter. **Note:** Control shall also be returned to the ECU if the ECU returns to **Normal/Default Diagnostic Session ($10 $81)**.

Hex	Diagnostic Tool Request Message
$30	**Input Output Control By Local Identifier Request**
$10	Input Output Local Identifier
$00	Input Output Control Parameter = Return Control to ECU

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$70	**Input Output Control By Local Identifier Positive Response**	**$7F**	**Negative Response Service Identifier**
$10	Input Output Local Identifier	$30	Input Output Control By Local Identifier Request
$00	Input Output Control Parameter = Return Control to ECU	$XX	Negative Response Code

Table 3.16.5-5 Implementation Example of IO Control Using Return Control to ECU Control Parameter

After the ECU resumes control from the diagnostic tester, the data in local identifier $10 should match those shown in Table 3.16.5-2.

3.17 SERVICE ID $31 – *START ROUTINE BY LOCAL IDENTIFIER*

3.17.1 MESSAGE DESCRIPTION

PURPOSE

The service, **Start Routine By Local Identifier ($31),** is used by the diagnostic tool to start a routine in an ECU's memory. The routines could either be tests that run instead of normal operating code or could be routines that are enabled and executed with the normal operating code running. In the first case, it might be necessary to switch the ECU to a specific Diagnostic Session using the **Start Diagnostic Session ($10)** service or to unlock the ECU using the **Security Access ($27)** service prior to using the **Start Routine By Local Identifier ($31)** service.

REQUIREMENTS

1. The ECU routine shall be started in the ECU's memory some time between the reception of the **Start Routine By Local Identifier Request ($31)** service and the completion of:
   a. **Start Routine By Local Identifier Positive Response ($71)** service
      -or if the routine cannot respond positively within the appropriate timing-
   b. **Start Routine By Local Identifier Negative Response ($7F)** service with **Request Correctly Received – Response Pending ($78)** negative response code.
2. If the routine needs a considerable amount of time to run to completion (e.g. more than 2s) it shall return immediately after the **Start Routine By Local Identifier Request ($31)** with a positive response. This positive response may contain Routine Status data to indicate that the routine needs a specific amount of time to run to completion. If under the above conditions the routine returns results the service **Request Routine Results By Local Identifier Service ($33)** may be used to obtain the results.
3. Routines shall not be used in place of **Read Data**, **I/O-Control**, or **Variant Coding**.
4. If this service is implemented by an ECU and will not end of its own accord, then **Stop Routine By Local Identifier ($32)** must also be supported.
5. If this service is implemented by an ECU and will return results, then either the **Start Routine By Local Identifier Positive Response ($71)** shall contain the resulting data or the **Request Routine Results By Local Identifier ($33)** must be supported.
6. The ECU shall exit any routine following a reset or powerdown event.

3.17.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$31**	**Start Routine By Local Identifier Request Service Id**	**Mandatory**	
1	**$XX**	**Routine Local Identifier**	**Mandatory**	**100**
	$00	Reserved	N/A	
	$01 - $DF	Routine Local Identifier	Optional	100
	$E0	Flash Erase Routine	Conditional[1]	100
	$E1	Flash Check Routine	Conditional[1]	100
	$E2	Reserved	N/A	
	$E3	Request DTCs From Shadow Error Memory	Optional	101
	$E4	Request Environment Data From Shadow Error Memory	Optional	101
	$E5	Request Event Information	Optional	102
	$E6	Request Event Environment Data	Optional	102
	$E7	Request Software Module Information	Conditional[2]	102
	$E8	Clear Tell-Tale Retention Stack	Conditional[3]	102
	$E9	Clear Event Information	Optional	102
	$EA-$F9	Reserved	N/A	
	$FA-FE	System Supplier Specific	N/A	105
	$FF	Reserved	N/A	
2	**$XX**	**Routine Entry Option #1**	**Optional**	**100**
:	**:**	**:**		
n	**$XX**	**Routine Entry Option #m**	**Optional**	**100**

Table 3.17.2-1 Start Routine By Local Identifier Request Message Format

Condition[1]: This Identification Option shall be supported according to the Flash Reprogramming Requirements Specification. (See Reference "*L"* in "Appendix A – *References*")
Condition[2]: Shall be supported if software module generation tool version ≥ 2.33 is used
Condition[3]: DCA ECU must support this feature.

3.17.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$71**	**Start Routine By Local Identifier Positive Response Service Id**	**Mandatory**	
1	**$XX**	**Routine Local Identifier** The **Routine Local Identifier** must be identical to the **Routine Local Identifier** sent in the request message. Refer to Table 3.17.2-1	**Mandatory**	**100**
2	**$XX**	**Routine Status #1**	**Optional**	**100**
:	**:**	**:**		
n	**$XX**	**Routine Status #m**	**Optional**	**100**

Table 3.17.3-1 Start Routine By Local Identifier Positive Response Message Format

3.17.4 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Mandatory**	
1	**$31**	**Start Routine By Local Identifier Request Service ID**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.17.4-1 Start Routine By Local Identifier Negative Response Message

3.17.5 IMPLEMENTATION EXAMPLE OF **START ROUTINE BY LOCAL IDENTIFIER** (RECEIVER TEST)

In this example an internal test routine is started that checks the functionality of a receiver. The positive response indicates whether the test was successful or not.

Hex	Diagnostic Tool Request Message
$31	**Start Routine By Local Identifier Request**
$01	Routine Local Identifier = Receiver Test

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$71	**Start Routine By Local Identifier Positive Response**	**$7F**	**Negative Response Service Identifier**
$01	Routine Local Identifier = Receiver Test	$31	Start Routine By Local Identifier Request
$01	Test Successful	$XX	Negative Response Code

Table 3.17.5-1 Implementation Example of Start Routine By Local Identifier

3.18 SERVICE ID $32 – *STOP ROUTINE BY LOCAL IDENTIFIER*

3.18.1 MESSAGE DESCRIPTION

PURPOSE

The service, **Stop Routine By Local Identifier ($32),** is used by the diagnostic tool to stop a routine in the ECU's memory initiated by **Start Routine By Local Identifier ($31)** service.

REQUIREMENTS

1. The ECU routine shall be stopped in the ECU's memory some time between the completion of the **Stop Routine By Local Identifier** Request ($32) service and the completion of:
   a. **Stop Routine By Local Identifier Positive Response ($72)** service or
   b. **Stop Routine By Local Identifier Negative Response ($7F)** service with **Request Correctly Received – Response Pending ($78)** negative response codes.
2. If this service is implemented by an ECU, then **Start Routine By Local Identifier ($31)** must also be supported.

3.18.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$32**	**Stop Routine By Local Identifier Request Service Id**	**Mandatory**	
1	**$XX**	**Routine Local Identifier**	**Mandatory**	**100**
	$00	Reserved	N/A	
	$01 - $DF	Routine Local Identifier	Optional	100
	$E0	Reserved	N/A	
	$E1	Reserved	N/A	
	$E2	Reserved	N/A	
	$E3	Request DTCs From Shadow Error Memory	Optional	101
	$E4	Request Environment Data From Shadow Error Memory	Optional	101
	$E5	Request Event Information	Optional	102
	$E6	Request Event Environment Data	Optional	102
	$E7	Reserved	N/A	
	$E8	Reserved	N/A	
	$E9	Reserved	N/A	
	$EA-$F9	Reserved	N/A	
	$FA-FE	System Supplier Specific	N/A	105
	$FF	Reserved	N/A	
2	**$XX**	**Routine Exit Option #1**	**Optional**	**100**
:	**:**	**:**	**:**	
n	**$XX**	**Routine Exit Option #m**	**Optional**	**100**

Table 3.18.2-1 Stop Routine By Local Identifier Request Message Format

3.18.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$72**	**Stop Routine By Local Identifier Positive Response Service Id**	**Mandatory**	
1	**$XX**	**Routine Local Identifier** The **Routine Local Identifier** must be identical to the **Routine Local Identifier** sent in the request message. Refer to Table 3.18.2-1	**Mandatory**	**100**
2	**$XX**	**Routine Exit Status**	**Optional**	**100**
	$00-$60	Reserved	N/A	
	$61	Normal Exit With Results Available	Optional	100
	$62	Normal Exit Without Results Available	Optional	100
	$63	Reserved	N/A	
	$64	Abnormal Exit Without Results Available	Optional	100
	$65-$FF	Reserved	N/A	

Table 3.18.3-1 Stop Routine By Local Identifier Positive Response Message Format

3.18.4 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Mandatory**	
1	**$32**	**Stop Routine By Local Identifier Request Service ID**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.18.4-1 Stop Routine By Local Identifier Negative Response Message

3.18.5 IMPLEMENTATION OF **STOP ROUTINE BY LOCAL IDENTIFIER** (INPUT/OUTPUT SIGNAL INTERMITTENT TEST ROUTINE)

This example specifies the test conditions to stop a routine in the ECU which has continuously tested (as fast as possible) all input and output signals on intermittence while a technician would have "wiggled" all wiring harness connectors of the system under test. The **Routine Local Identifier** references this routine by the **Routine Local Identifier ($01)**. In this example, the routine needs to be stopped with **Stop Routine By Local Identifier ($32)** because the routine will not end on its own.

Test conditions:

1. Ignition = On
2. Engine = Off
3. Vehicle speed = 0 [mph]

Hex	Diagnostic Tool Request Message
$32	**Stop Routine By Local Identifier Request**
$01	Routine Local Identifier = Input Output Signal Intermittent Test Routine

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$72	**Start Routine By Local Identifier Positive Response**	**$7F**	**Negative Response Service Identifier**
$01	Routine Local Identifier = Input Output Signal Intermittent Test Routine	$32	Stop Routine By Local Identifier Request
		$XX	Negative Response Code

Table 3.18.5-1 Implementation Example of Stop Routine By Local Identifier

3.19 SERVICE ID $33 – *REQUEST ROUTINE RESULTS BY LOCAL IDENTIFIER*

3.19.1 MESSAGE DESCRIPTION

PURPOSE

The service, **Request Routine Results By Local Identifier ($33),** is used by the diagnostic tool to request results (e.g. exit status information) referenced by a **Routine Local Identifier**.

3.19.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$33**	**Request Routine Results By Local Identifier Request Service Id**	**Mandatory**	
1	**$XX**	**Routine Local Identifier**	**Mandatory**	100
	$00	Reserved	N/A	
	$01 - $DF	Routine Local Identifier	Optional	100
	$E0	Flash Erase Routine	Conditional[1]	100
	$E1	Flash Check Routine	Conditional[1]	100
	$E2	Tell-Tale Retention Stack	Conditional[2]	100
	$E3	Request DTCs From Shadow Error Memory	Optional	101
	$E4	Request Environment Data From Shadow Error Memory	Optional	101
	$E5	Request Event Information	Optional	102
	$E6	Request Event Environment Data	Optional	102
	$E7	Reserved	N/A	
	$E8	Reserved	N/A	
	$E9	Reserved	N/A	
	$EA-$F9	Reserved	N/A	105
	$FA-FE	System Suppler Specific	N/A	105
	$FF	Reserved	N/A	

Table 3.19.2-1 Request Routine Results By Local Identifier Request Message Format

Condition[1]: This Identification Option shall be supported according to the Flash Reprogramming Requirements Specification. (See Reference "*L"* in "Appendix A – *References*")
Condition[2]: Returning the Tell-Tale Retention Stack results may be supported by DCA ECU's.

3.19.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$73**	**Request Routine Results By Local Identifier Positive Response Service Id**	**Mandatory**	
1	**$XX**	**Routine Local Identifier** The **Routine Local Identifier** must be identical to the **Routine Local Identifier** sent in the request message. Refer to Table 3.19.2-1	**Mandatory**	**100**
2	**$XX**	**Routine Results #1**	**Mandatory**	**102**
:	**:**	**:**		
n	**$XX**	**Routine Results #m**	**Optional**	**102**

Table 3.19.3-1 Request Routine Results By Local Identifier Positive Response Message Format

3.19.4 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Mandatory**	
1	**$33**	**Request Routine Results By Local Identifier Request Service ID**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.19.4-1 Request Routine Results By Local Identifier Negative Response Message

3.20 SERVICE ID $34 – REQUEST DOWNLOAD

3.20.1 MESSAGE DESCRIPTION

PURPOSE

The service, **Request Download ($34)**, is used by the diagnostic tool to initiate a data transfer from the requester to the receiver. (i.e. Diagnostic tool to ECU.) This service is typically used in Flash Reprogramming procedures.

REQUIREMENTS

1) After an ECU has received the **Request Download Request ($34)** service, the ECU shall make all necessary preparations to receive data before it sends a **Request Download Positive Response ($74)** service.
2) Any ECU that performs a **Request Download** shall implement appropriate security mechanisms.

3.20.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$34**	**Request Download Request Service Id**	**Mandatory**	
1	**$XX**	**Memory Address (High Byte)**	**Mandatory**	**98**
2	**$XX**	**Memory Address (Middle Byte)**	**Mandatory**	**98**
3	**$XX**	**Memory Address (Low Byte)**	**Mandatory**	**98**
4	**$00**	**Data Format Identifier**	**Mandatory**	**102**
	$0x	Uncompressed Data	Mandatory	103
	$1x - $Fx	Compressed Data	Optional	102
	$x0	Unencrypted Data	Mandatory	103
	$x1 - $xF	Encrypted Data	Optional	103
5	**$XX**	**Uncompressed Memory Size { High Byte }**	**Mandatory**	**104**
6	**$XX**	**Uncompressed Memory Size { Middle Byte }**	**Mandatory**	**104**
7	**$XX**	**Uncompressed Memory Size { Low Byte }**	**Mandatory**	**104**

Table 3.20.2-1 Request Download Request Message Format

3.20.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$74**	**Request Download Positive Response Service Id**	**Mandatory**	
1	**$XX**	**Max number Of Block Length { High Byte }**	**Mandatory**	**103**
2	**$XX**	**Max number Of Block Length { Low Byte }**	**Optional**	

Table 3.20.3-1 Request Download Positive Response Message Format

3.20.4 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response Message**	**Mandatory**	
1	**$34**	**Request Download Request Service ID**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.20.4-1 Request Download Negative Response Message

3.20.5 IMPLEMENTATION EXAMPLE OF **REQUEST DOWNLOAD**

Refer to 3.23.5 on page 60 for a complete example of **Request Download ($34)** including **Data Transfer ($36)** and **Data Transfer Exit ($37)**.

3.21 SERVICE ID $35 – *REQUEST UPLOAD*

3.21.1 MESSAGE DESCRIPTION

PURPOSE

The service, **Request Upload ($35),** is used by the diagnostic test tool to initiate a data transfer from the source to the requester. (i.e. ECU to diagnostic tool.) This service is typically used in Flash Reprogramming procedures.

REQUIREMENTS

1. After the ECU has received the **Request Upload Request ($35)** service, the ECU shall make all necessary preparations to send data before it sends a **Request Upload Positive Response ($75)**.

3.21.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$35**	**Request Upload Request Service Id**	**Mandatory**	
1	**$XX**	**Memory Address (High Byte)**	**Mandatory**	**98**
2	**$XX**	**Memory Address (Middle Byte)**	**Mandatory**	**98**
3	**$XX**	**Memory Address (Low Byte)**	**Mandatory**	**98**
4	**$00**	**Data Format Identifier**	**Mandatory**	**102**
	$0x	Uncompressed Data	Mandatory	103
	$1x - $Fx	Compressed Data	Optional	102
	$x0	Unencrypted Data	Mandatory	103
	$x1 - $xF	Encrypted Data	Optional	103
5	**$XX**	**Uncompressed Memory Size { High Byte }**	**Mandatory**	**104**
6	**$XX**	**Uncompressed Memory Size { Middle Byte }**	**Mandatory**	**104**
7	**$XX**	**Uncompressed Memory Size { Low Byte }**	**Mandatory**	**104**

Table 3.21.2-1 Request Upload Request Message Format

3.21.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$75**	**Request Upload Positive Response Service Id**	**Mandatory**	
1	**$XX**	**Max Number Of Block Length { High Byte }**	**Mandatory**	**103**
2	**$XX**	**Max number Of Block Length { Low Byte }**	**Optional**	

Table 3.21.3-1 Request Upload Positive Response Message Format

3.21.4 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Mandatory**	
1	**$35**	**Request Upload Request Service ID**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.21.4-1 Request Upload Negative Response Message

3.21.5 IMPLEMENTATION EXAMPLE OF **REQUEST UPLOAD**

Refer to Section 3.20.5 on page 60 for a complete example of **Request Upload ($35)** including **Data Transfer ($36)** and **Data Transfer Exit ($37)**.

3.22 SERVICE ID $36 – *TRANSFER DATA*

3.22.1 MESSAGE DESCRIPTION

PURPOSE

The service, **Transfer Data ($36)**, is used by the diagnostic tool to transfer data. The direction of the data transfer is defined by the preceding **Request Upload ($35)** service or **Request Download ($34)** service. This service is typically used in Flash Reprogramming procedures.

REQUIREMENTS

1) If **Request Download ($34)** service precedes **Transfer Data ($36)** service, the data to be downloaded shall be included in the **Transfer Request Parameter(s)** in the **Transfer Data Request ($36)** service(s).
2) If **Request Upload ($35)** service precedes **Transfer Data ($36)** service, the data to be uploaded shall be included in the **Transfer Response Parameter(s)** in the **Transfer Data Positive Response ($76)** service(s).

3.22.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$36**	**Transfer Data Request Service Id**	**Mandatory**	
1	**$XX**	**Transfer Data Request Parameter #1 / Block Sequence Counter**	**Conditional[1]**	**103 / 103**
2	**$XX**	**Transfer Data Request Parameter #2 / Transfer Data Request Parameter #1**	**Mandatory**	**103**
:	**:**	**:**		
n	**$XX**	**Transfer Data Request Parameter #m / Transfer Data Request Parameter #m+1**	**Optional**	**103**

Table 3.22.2-1 Transfer Data Request Message Format

Conditional[1]: If the Block Sequence Counter is used it shall be the first byte following the Request ID. If the Block Sequence Counter is not used, the first byte following the Request ID shall be the Transfer Data Request Parameter. An ECU should use $21 $E4 to indicate the whether or not the Block Sequence Counter is used. (Refer to Reprogramming Fault Reporting ($E4) on p. 94 for more information.)

3.22.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$76**	**Transfer Data Positive Response Service Id**	**Mandatory**	
1	**$XX**	**Transfer Data Request Parameter #1 / Block Sequence Counter**	**Conditional[1]**	**103**
2	**$XX**	**Transfer Data Response Parameter #1**	**Mandatory**	**103**
:	**:**	**:**		
n	**$XX**	**Transfer Data Response Parameter #m**	**Optional**	**103**

Table 3.22.3-1 Transfer Data Positive Response Message Format

Conditional[1]: If the Block Sequence Counter is used it shall be the first byte following the Request ID. If the Block Sequence Counter is not used, the first byte following the Request ID shall be the Transfer Data Request Parameter. An ECU should use $21 $E4 to indicate the whether or not the Block Sequence Counter is used. (Refer to Reprogramming Fault Reporting ($E4) on p. 94 for more information.)

3.22.4 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Response Message**	**Mandatory**	
1	**$36**	**Transfer Data Request Service ID**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.22.4-1 Transfer Data Negative Response Message

3.22.5 IMPLEMENTATION EXAMPLE OF **TRANSFER DATA**

Refer to 3.23.5 on page 60 for an example of **Data Transfer ($36)** including **Request Upload ($35)** and **Data Transfer Exit ($37)**.

3.23 SERVICE ID $37 – *REQUEST TRANSFER EXIT*

3.23.1 MESSAGE DESCRIPTION

PURPOSE

The service, **Request Transfer Exit ($37)**, is used to indicate a data transfer has been completed. This service is typically used in Flash Reprogramming procedures.

REQUIREMENTS

3.23.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	$37	Request Transfer Exit Request Service Id	Mandatory	
1	$XX	Transfer Data Request Parameter #1	Optional	103
:	:	:		
n	$XX	Transfer Data Request Parameter #m	Optional	103

Table 3.23.2-1 Request Transfer Exit Request Message Format

3.23.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	$77	Request Transfer Exit Positive Response Service Id	Mandatory	
1	$XX	Transfer Data Response Parameter #1	Optional	103
:	:	:		
n	$XX	Transfer Data Response Parameter #m	Optional	103

Table 3.23.3-1 Request Transfer Positive Response Message Format

3.23.4 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	$7F	Negative Response	Mandatory	
1	$37	Request Transfer Exit Request Service ID	Mandatory	
2	$XX	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	Mandatory	

Table 3.23.4-1 Request Transfer Negative Response Message

3.23.5 IMPLEMENTATION EXAMPLE OF **REQUEST TRANSFER**

EXAMPLE #1 IMPLEMENTATION OF REQUEST DOWNLOAD, TRANSFER DATA, AND REQUEST TRANSFER EXIT

This section specifies the conditions to transfer (download) data from the diagnostic tool to the ECU.
The example consists of 3 steps:

1) Request a download (**Request Download ($34)**)
2) Transfer the data (**Transfer Data ($36)**)
   The diagnostic tool transfers 64K Bytes of data to the flash memory in the ECU starting at Memory Address $602000.
3) Exit the data transfer (**Request Transfer Exit ($37)**)
   The diagnostic tool indicates the data transfer has completed to the ECU with a Request Transfer Exit service.

Hex	Diagnostic Tool Request Message
$34	Request Download Request
$60	Memory Address (High Byte)
$20	Memory Address (Middle Byte)
$00	Memory Address (Low Byte)
$00	Data Format Identifier (No compression, No encryption)
$00	Uncompressed Memory Size (High Byte)
$FA	Uncompressed Memory Size (Middle Byte)
$00	Uncompressed Memory Size (Low Byte)

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$74	Request Download Positive Response	$7F	Negative Response Service Identifier
$81	Max Number of Block Length	$34	Request Download Request
		$XX	Negative Response Code

Hex	Diagnostic Tool Request Message
$36	Transfer Data Request
$XX	Transfer Request Parameter #1 = Transfer Data Byte #1 (Byte 2)
:	:
$XX	Transfer Request Parameter #128 = Transfer Data Byte #128 (Byte 129)

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$76	Transfer Data Positive Response	$7F	Negative Response Service Identifier
		$36	Transfer Data Request
		$XX	Negative Response Code

Hex	Diagnostic Tool Request Message
$36	Transfer Data Request
$XX	Transfer Request Parameter #1 = Data Byte #2
:	:
$XX	Transfer Request Parameter #n-1 = Data Byte #n

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$76	Transfer Data Positive Response	$7F	Negative Response Service Identifier
		$36	Transfer Data Request
		$XX	Negative Response Code

Hex	Diagnostic Tool Request Message
$37	Request Transfer Exit Request

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$77	Request Transfer Exit Positive Response	$7F	Negative Response Service Identifier
		$37	Request Transfer Exit Request
		$XX	Negative Response Code

Table 3.23.5-1 Implementation Example of Request Download, Transfer Data, and Request Transfer Exit Request

EXAMPLE #2 IMPLEMENTATION OF REQUEST UPLOAD, TRANSFER DATA, AND REQUEST TRANSFER EXIT

This section specifies the conditions to transfer data (upload) from an ECU to the diagnostic tool.
The example consists of 3 steps:

1) Request an upload **(Request Upload ($35))**
2) Transfer the data **(Transfer Data ($36))**
   The ECU transfers 511 data bytes using 3 **Transfer Data** services with 129 data bytes (128 ECU data bytes + 1 service ID data byte) and 1 **Transfer Data** service with 128 data bytes (127 ECU data bytes + 1 service ID data byte) starting at **Memory Address** $201000 in the ECU.
3) Exit the data transfer **(Request Transfer Exit ($37))**
   The diagnostic tool indicates the data transfer from the ECU has completed with a **Request Transfer Exit** service.

Hex	Diagnostic Tool Request Message
$35	**Request Upload Request**
$20	Memory Address (High Byte)
$10	Memory Address (Middle Byte)
$00	Memory Address (Low Byte)
$00	Data Format Identifier (No compression, No encryption)
$00	Uncompressed Memory Size (High Byte)
$01	Uncompressed Memory Size (Middle Byte)
$FF	Uncompressed Memory Size (Low Byte)

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$75	**Request Upload Positive Response**	**$7F**	**Negative Response Service Identifier**
$81	Max Number Of Block Length	$35	Request Upload Request
		$XX	Negative Response Code

Hex	Diagnostic Tool Request Message
$36	**Transfer Data Request (#1)**

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$76	**Transfer Data Positive Response**	**$7F**	**Negative Response Service Identifier**
$XX	Transfer Request Parameter #1 = Transfer Data Byte #1 (Byte #2)	$36	**Transfer Data Request**
:	:	$XX	Negative Response Code
$XX	Transfer Request Parameter #128 = Transfer Data Byte #128 (Byte #129)		

Hex	Diagnostic Tool Request Message
$36	**Transfer Data Request (#2)**

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$76	**Transfer Data Positive Response**	**$7F**	**Negative Response Service Identifier**
$XX	Transfer Request Parameter #1 = Transfer Data Byte #1 (Byte #2)	$36	**Transfer Data Request**
:	:	$XX	Negative Response Code
$XX	Transfer Request Parameter #128 = Transfer Data Byte #128 (Byte #129)		

Table 3.23.5-2 Implementation Example of Request Upload, Transfer Data, and Request Transfer Exit Request – Part A

Hex	Diagnostic Tool Request Message
$36	**Transfer Data Request (#3)**

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$76	**Transfer Data Positive Response**	**$7F**	**Negative Response Service Identifier**
$XX	Transfer Request Parameter #1 = Transfer Data Byte #1 (Byte #2)	$36	**Transfer Data Request**
:	:	$XX	Negative Response Code
$XX	Transfer Request Parameter #128 = Transfer Data Byte #128 (Byte #129)		

Hex	Diagnostic Tool Request Message
$36	**Transfer Data Request (#4)**

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$76	**Transfer Data Positive Response**	**$7F**	**Negative Response Service Identifier**
$XX	Transfer Request Parameter #1 = Transfer Data Byte #1 (Byte #2)	$36	**Transfer Data Request**
:	:	$XX	Negative Response Code
$XX	Transfer Request Parameter #128 = Transfer Data Byte #127 (Byte #128)		

Hex	Diagnostic Tool Request Message
$37	**Request Transfer Exit Request**

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$77	**Request Transfer Exit Positive Response**	**$7F**	**Negative Response Service Identifier**
		$37	Request Transfer Exit Request
		$XX	Negative Response Code

Table 3.23.5-3 Implementation Example of Request Upload, Transfer Data, and Request Transfer Exit Request – Part B

3.24 SERVICE ID $3B – *WRITE DATA BY LOCAL IDENTIFIER*

3.24.1 MESSAGE DESCRIPTION

PURPOSE

The service, **Write Data By Local Identifier ($3B),** is used by the diagnostic tool to write **Record Values** (data values) to an ECU. The **Local Identifier** data is read by using the **Record Local Identifier ($21)** or **Read ECU Identification ($1A)** (LID Range $80-$9F).

REQUIREMENTS

Writing DCS-Variant Coding shall be performed with **Write Data By Local Identifier ($3B).**

3.24.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$3B**	**Write Data By Local Identifier Request Service Id**	**Mandatory**	
1	**$XX**	**Record Local Identifier**	**Mandatory**	**85**
	$00	Reserved	N/A	
	$01-$7F	Record Local Identifier	Optional	85
	$86	DCS ECU Identification	Optional	86
	$87	DCX / MMC ECU Identification (Note: DCX = DCA + DCS)	Optional	87
	$88	VIN (Orginal)	Conditional[1]	88
	$89	Diagnostic Variant Code	Optional	89
	$8A - $8F	System Supplier Specific	N/A	105
	$90	VIN (Current)	Conditional[1]	89
	$96	Calibration Identification	Optional	89
	$97	Calibration Verification Identification	Optional	90
	$9A	ECU Code Fingerprint	Conditional[2]	90
	$9B	ECU Data Fingerprint	Conditional[2]	91
	$9C	ECU Code Software Identification	Optional	91
	$9D	ECU Data Software Identification	Optional	92
	$9E	ECU Boot Software Identification	Optional	92
	$9F	ECU Boot Fingerprint	Conditional[2]	92
	$A0-$DF	Record Local Identifier	Optional	85
	$E0	Development Data	Optional	92
	$E1	ECU Serial Number	Optional	93
	$E2	DBCom Data	Optional	93
	$E3	Operating System Version	Optional	94
	$E4	*Reserved*	N/A	
	$E5	Vehicle Information	Optional	94
	$E6	Flash Info 1 (read only)	N/A	95
	$E7	Flash Info 2 (read only)	N/A	95
	$E8	Systemdiagnostic general parameter data (read only)	N/A	95
	$E9	Systemdiagnostic global parameter data (read only)	N/A	96
	$EA-$EF	Reserved	N/A	105
	$F0-$F9	Dynamically Defined Local Identifiers	N/A	97
	$FA-$FE	System Supplier Specific	N/A	105
	$FF	Reserved	N/A	
2	**$XX**	**Record Value #1**	**Mandatory**	**98**
:	**:**			
n	**$XX**	**Record Value #m**	**Conditional[3]**	**98**

Table 3.24.2-1 Write Data By Local Identifier Request Message Format

Condition[1]: This Identification Option may be supported by ECUs that are used at DCA. For ECU's used at DCS, this local identifier is optional.
Condition[2]: This Identification Option shall be supported according to the Flash Reprogramming Requirements. (See Reference "*L"* in "Appendix A – References")
Condition[3]: Record Value #m shall be included in the request if data length exceeds 1 byte.

3.24.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7B**	**Write Data By Local Identifier Positive Response Service Id**	**Mandatory**	
1	**$XX**	**Record Local Identifier** The **Record Local Identifier** must be identical to the **Record Local Identifier** sent in the request message. Refer to Table 3.24.2-1	**Mandatory**	**85**

Table 3.24.3-1 Write Data By Local Identifier Positive Response Message Format

3.24.4 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Mandatory**	
1	**$3B**	**Write Data By Local Identifier Request Service ID**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.24.4-1 Write Data By Local Identifier Negative Response Message

3.24.5 IMPLEMENTATION EXAMPLE OF **WRITE DATA BY LOCAL IDENTIFIER**

The diagnostic tool shall write the VIN data to the ECU by using the **Write Data By Local Identifier** service. The ECU shall send a **Write Data By Local Identifier Positive Response ($7B)** message after it has successfully programmed the data.

Hex	Diagnostic Tool Request Message
$3B	**Write Data By Local Identifier Request**
$90	Record Local Identifier = Identification Option (VIN)
$57	VIN……..(W)
$30	VIN……..(0)
$4C	VIN……..(L)
$30	VIN……..(0)
$30	VIN……..(0)
$30	VIN……..(0)
$30	VIN……..(0)
$34	VIN……..(4)
$33	VIN……..(3)
$3D	VIN……..(M)
$42	VIN……..(B)
$35	VIN……..(5)
$34	VIN……..(4)
$31	VIN……..(1)
$33	VIN……..(3)
$32	VIN……..(2)
$36	VIN……..(6)

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$7B	**Write Data By Local Identifier Positive Response**	**$7F**	**Negative Response Service Identifier**
$90	Record Local Identifier = Identification Option (VIN)	$3B	Write Data By Local Identifier Request
		$XX	Negative Response Code

Table 3.24.5-1 Implementation Example of Write Data By Local Identifier

3.25 SERVICE ID $3D – *WRITE MEMORY BY ADDRESS*

3.25.1 MESSAGE DESCRIPTION

PURPOSE

The service, **Write Memory By Address ($3D),** is used by the diagnostic tool to write **Record Values** (data values) to an ECU. The data are identified by the ECU's **Memory Address** and **Memory Size**.

REQUIREMENTS

1) The service **Write Memory By Address ($3D)** shall only be used during the period of ECU development. The service **Write Memory By Local ID ($3B)** shall be used in production and service.
2) The **Write Memory By Address Positive Response ($7D)** service shall always return the **Memory Address** parameter specified in the **Write Memory By Address Request ($3D)** service.
3) The **Write Memory By Address Negative Response ($7F)** service shall always return an appropriate negative response code when the **Write Memory By Address Request ($3D)** service cannot be fulfilled.
4) Any ECU that must have access to **Memory Addresses** (e.g. for debug-information) shall implement appropriate security mechanisms.

3.25.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$3D**	**Write Memory By Address Request Service Id**	**Mandatory**	
1	**$XX**	**Memory Address (High Byte)**	**Mandatory**	**98**
2	**$XX**	**Memory Address (Middle Byte)**	**Mandatory**	**98**
3	**$XX**	**Memory Address (Low Byte)**	**Mandatory**	**98**
4	**$XX**	**Memory Size**	**Mandatory**	**98**
5	**$XX**	**Record Value #1**	**Mandatory**	**98**
:	**:**	**:**		
n	**$XX**	**Record Value #m**	**Conditional**	**98**

Table 3.25.2-1 Write Memory By Address Request Message Format

Condition: Record Value #m shall be included in the request if Memory Size exceeds 1 byte.

3.25.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7D**	**Write Memory By Address Positive Response Service Id**	**Mandatory**	
1	**$XX**	**Memory Address (High Byte)**	**Mandatory**	**98**
2	**$XX**	**Memory Address (Middle Byte)**	**Mandatory**	**98**
3	**$XX**	**Memory Address (Low Byte)**	**Mandatory**	**98**

Table 3.25.3-1 Write Memory By Address Positive Response Message Format

3.25.4 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Mandatory**	
1	**$3D**	**Write Memory By Address Request Service ID**	**Mandatory**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Mandatory**	

Table 3.25.4-1 Write Memory By Address Negative Response Message

3.25.5 IMPLEMENTATION EXAMPLE OF **WRITE MEMORY BY ADDRESS**

The ECU writes ten (10) data bytes to the ECU's serial EEPROM at the memory address $30FF13.

Hex	Diagnostic Tool Request Message
$3D	**Write Memory Address Request**
$30	Memory Address (High Byte)
$FF	Memory Address (Middle Byte)
$13	Memory Address (Low Byte)
$0A	Memory Size
$11	Record Value #1
$22	Record Value #2
$33	Record Value #3
$44	Record Value #4
$55	Record Value #5
$66	Record Value #6
$77	Record Value #7
$88	Record Value #8
$99	Record Value #9
$AA	Record Value #10

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$7D	**Write Memory Address Positive Response**	**$7F**	**Negative Response Service Identifier**
$30	Memory Address (High Byte)	$3D	Write Memory Address Request
$FF	Memory Address (Middle Byte)	$XX	Negative Response Code
$13	Memory Address (Low Byte)		

Table 3.25.5-1 Implementation Example of Write Memory By Address

3.26 SERVICE ID $3E – *TESTER PRESENT*

3.26.1 MESSAGE DESCRIPTION

PURPOSE:
The service, **Tester Present ($3E)**, is used to indicate to an ECU that the diagnostic tool is present. The Tester Present is used in the absence of other diagnostic communication to maintain **ECU Flash Re-programming Session ($10 $85), Stand By Session ($10 $89)**, and **Extended Diagnostic Session ($10 $92)**.

REQUIREMENTS:
1. If a time out occurs (P3max), the ECU has to return to the **Normal/Default Session ($81)**.

3.26.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$3E**	**Tester Present Request Service Id**	**Mandatory**	
1	**$XX**	**Response Required**	**Mandatory**	**99**
	$01	Response Required (Default)	Mandatory	99
	$02	No Response Required	Mandatory	99

Table 3.26.2-1 Tester Present Request Message Format

3.26.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7E**	**Tester Present Positive Response Service Id**	**Conditional**	

Table 3.26.3-1 Tester Present Positive Response Message Format

Condition: An ECU shall not return a **Tester Present Positive Response ($7E)** if the **No Response Required ($02)** parameter was used by the diagnostic test tool in the request.

3.26.4 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Conditional**	
1	**$3E**	**Tester Present Request Service ID**	**Conditional**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Conditional**	

Table 3.26.4-1 Tester Present Negative Response Message

Condition: An ECU shall not return a **Negative Response ($7F)** if the **No Response Required ($02)** parameter was used by the diagnostic test tool in the request.

3.27 SERVICE ID $85 – *CONTROL DTC (DIAGNOSTIC TROUBLE CODE) SETTING*

3.27.1 MESSAGE DESCRIPTION

PURPOSE:

The service, **Control DTC Setting ($85)** shall be used by a diagnostic test tool to enable and disable the setting of Diagnostic Trouble Codes, DTCs, in the ECU(s).

REQUIREMENTS:

1. Conditions for controlling DTC setting are as follows:
   i. Switch off setting of DTCs:

      - through diagnostic service **Control DTC Setting ($85)**, DTC setting mode "off"

   ii. Switch setting of DTCs on again:

      - through diagnostic service **Control DTC Setting ($85)**, DTC setting mode "on"
      - after loss of ECU supply voltage (reset or powerdown event)
      - when switching from the Extended Diagnostic Session to any other Diagnostic Session, especially Default Session.
2. Services Read DTC ($18), Read Status If DTC ($17) and Clear Diagnostic Information ($14) shall not be influenced by service Control DTC Setting ($85)

3.27.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$85**	**Control DTC Setting Request Service ID**	**Mandatory**	
1	**$XX**	**Response Required**	**Mandatory**	**99**
	$01	Response Required	Mandatory	99
	$02	No Response Required	Mandatory	99
2	**$XX**	**Group Of DTC { High Byte }**	**Mandatory**	**82**
3	**$XX**	**Group Of DTC { Low Byte }**	**Mandatory**	**82**
	$0000	All Powertrain DTCs	Optional	82
	$4000	All Chassis DTCs	Optional	82
	$8000	All Body DTCs	Optional	82
	$C000	All Network DTCs	Optional	82
	$FF00	All DTCs	Mandatory	83
4	**$XX**	**DTC Setting Mode**	**Mandatory**	**105**
	$00	Reserved	N/A	
	$01	On	Mandatory	105
	$02	Off	Mandatory	105
	$03-$FF	Reserved	N/A	

Table 3.27.2-1 Control DTC Setting Request Message Format

3.27.3 POSITIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$C5**	**Control DTC Setting Positive Response Service ID**	**Conditional**	

Table 3.27.3-1 Control DTC Setting Positive Response Message Format

Condition: An ECU shall not return a **Control DTC Setting Positive Response ($C5)** if the **No Response Required ($02)** parameter was used by the diagnostic test tool in the request.

3.27.4 NEGATIVE RESPONSE MESSAGE

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Conditional**	
1	**$85**	**Control DTC Setting Request Service ID**	**Conditional**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Conditional**	

Table 3.27.4-1 Control DTC Setting Transmission Negative Response Message

Condition: An ECU shall not return a **Negative Response ($7F)** if the **No Response Required ($02)** parameter was used by the diagnostic test tool in the request.

3.27.5 IMPLEMENTATION EXAMPLE OF **CONTROL DTC SETTING**

The following example shows the message flow if the ECU is in normal operating condition and the tester switches off the setting of DTCs.

Hex	Diagnostic Tool Request Message
$85	**Control DTC Setting Request Service ID**
$01	Response Required
$FF	All DTCs (MSB)
$00	All DTCs (MSB)
$02	Off

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$C5	**Control DTC Setting Positive Response**	**$7F**	**Negative Response Service Identifier**
		$85	Control DTC Setting
		$XX	Negative Response Code

Table 3.27.5-1 Implementation Example of Control DTC Setting

3.28 SERVICE ID $86 – RESPONSE ON EVENT

3.28.1 MESSAGE DESCRIPTION

PURPOSE:
The service, **Response On Event ($86),** requests an ECU to start or stop transmission of responses on a specified event. This service provides the possibility to trigger a diagnostic service when the specified event within the ECU occurs. The diagnostic tool specifies the event and the corresponding service to be executed when the event occurs.

An event may be defined as any change in data that the ECU has access to. For example, events include timer interrupts, fault setting, message on data link, or a threshold criteria being met. Within the Response On Event request a Service to Respond is defined. Every time the specified event occurs the positive response to this service is sent by the ECU.

See the figure below for a brief overview about the client and server behavior.

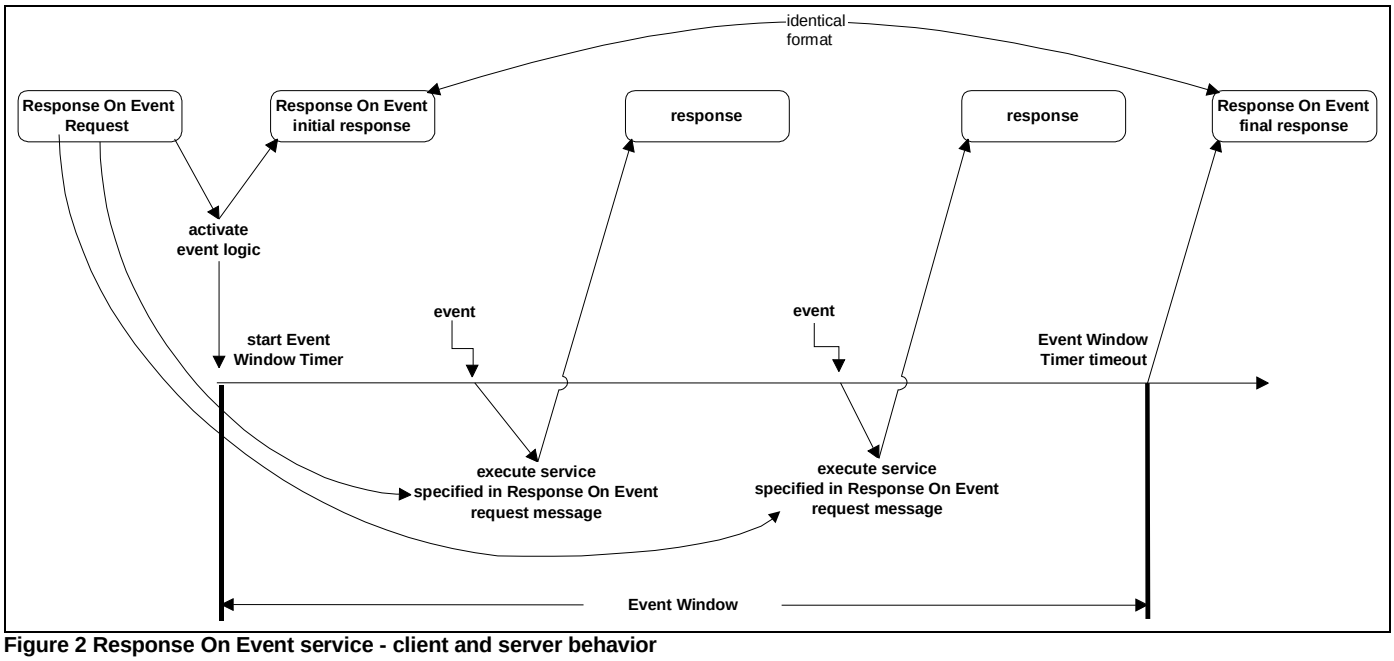

Figure 2 Response On Event service - client and server behavior

NOTE: The figure above assumes, that the event window timer is configured to timeout prior to the power down of the ECU, therefore the final **Response On Event** positive response message is shown at the end of the event timing window.

REQUIREMENTS:
1) Multiple **Response On Event** services may run concurrently with different requirements (different **Event Types**, **Service To Respond To** - **Records**...).

2) While the **Response On Event** service is active, the ECU must still be able to process diagnostic request and response messages accordingly. This should be accomplished with a (different) pair of CAN identifiers for the **Service To Respond To.** **Note**: At least the response CAN identifier shall be different.

   If the same diagnostic request/response CAN identifiers are used for diagnostic communication and the **Service To Respond To** – responses, the following restrictions shall apply:

   If a Response On Event service has been established with an ECU (e.g Request A) and the event specified by the Event Type occurs, the ECU shall ignore incoming diagnostic requests (e.g Request B) until the **Service To Respond To** - response is completed. Therefore the tester shall repeat the request (i.e Request B) according to requirement 2) 0 and 2) 0.

If Response On Event has been established (e.g with Service to respond to Request A) and the tester receives any response (e.g Response A or B), the tester has to distinguish between a **Service To Respond To** – response (e.g Response A) and the response to the request. (e.g Response B)

If the tester receives a **Service To Respond To** - response (e.g Response A) (one of the possible responses set up with **Response On Event** - service), it shall repeat the request (e.g Request B) after the **Service To Respond To**-response (e.g Response A) has been received completely.

If the response is ambiguous (e.g Response A or B) (i.e. the response could originate from the **Service To Respond To** (e.g Response A),initiated by an event, or from the response to a diagnostic request(e.g Response B),), the tester shall present the response both as a **Service To Respond To** response (e.g Response A), and as the response to the diagnostic request (e.g Response B),. The tester shall not repeat the request with the exception of **Negative Response Code - Busy Repeat Request ($21).**

Only Single Frame diagnostic requests shall be used during active **Response On Event** services if the diagnostic test tool and ECU are not on the same network (i.e. if there could be additional latencies between sending a frame and the addressee receiving it). (e.g. the message is routed through at least one gateway.) Refer to Figure 3.

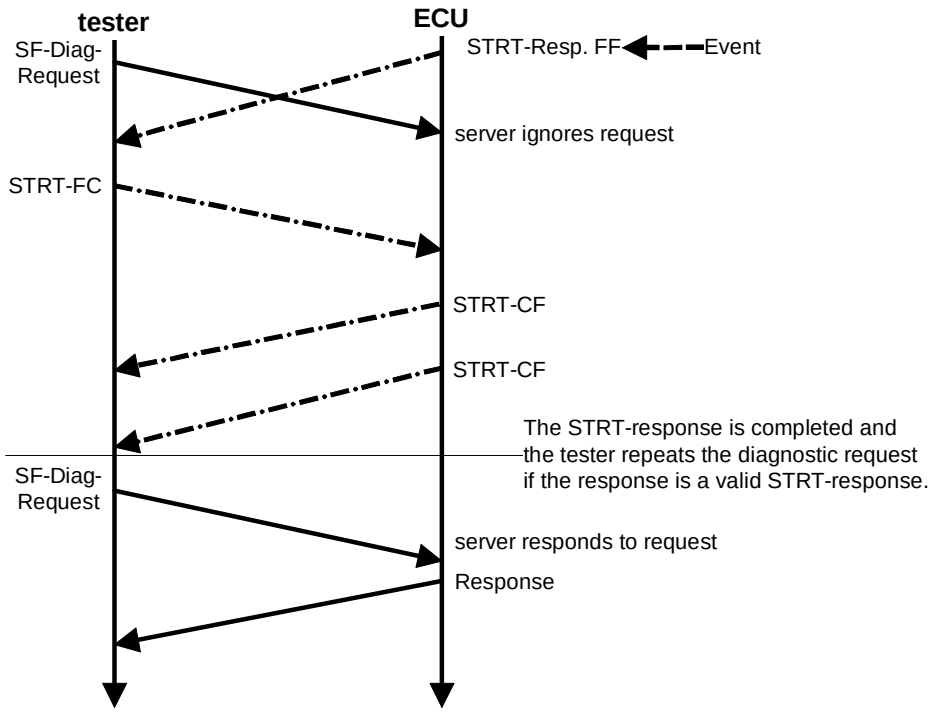

Figure 3 - Concurrent request when the event occurs

3) The **Response On Event** service shall only be allowed to use those diagnostic services available in the active diagnostic session.

4) While the **Response On Event** service is active, any change in a diagnostic session shall terminate the current **Response On Event** service(s). For instance, if a **Response On Event** service has been set up during **Extended Diagnostic Session**, it shall terminate when the ECU switches to the **Default Session**.

5) After receiving the **Response On Event** request message, the ECU shall respond with an initial positive response acknowledgement after opening the event window. This acknowledgement indicates that the **Response On Event** service has been enabled.

6) The sub-function parameter value **Response Required** = "No" should only be used for the **Event Type** = **Stop Response On Event**. The ECU shall always return a response to the event-triggered response when the specified event is detected.

7) The ECU shall return a final positive response to indicate the **Response On Event** ($86) service has reached the end of the event window unless one of the following conditions apply:

   — if **Event Types** do not start **Response On Event**, such as **Stop Response On Event** or **Report Activated Events**

   — if the Event Window = Infinite event was established

   — if the Service has been deactivated before the event window was closed (e.g The ECU switches to Default Session)

8) When the specified event is detected, the ECU shall respond immediately with the appropriate Service To Respond To - response message. However, the immediate Service To Respond To - response message shall not disrupt any other diagnostic request or response transmission already in progress (i.e. the Service To Respond To - response must be delayed until the current message transmission has been completed) Refer to Figure 4.

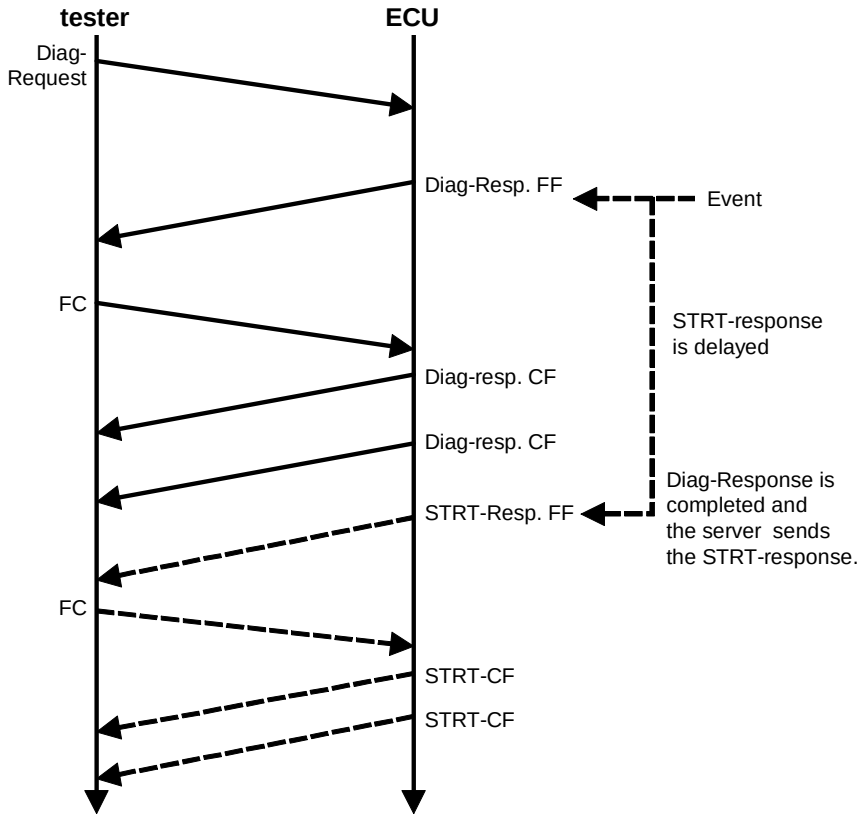

Figure 4 - Event occurrence during a message being in progress

9) The Response On Event service shall only apply to transient events and conditions. The ECU shall return a response once per event occurrence. For a condition that is continuously sustained over a period of time, the response service shall be executed only one time at the initial occurrence. In case the Event Type is defined so that Service To Respond To responses could occur at a high frequency then appropriate measures have to be taken in order to prevent back to back Service To Respond To responses. A minimum separation time between Service To Respond To responses could be part of the Event Type Record (vehicle manufacturer specific).

3.28.2 REQUEST MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$86**	**Response On Event Request Service Id**	**Mandatory**	
1	**$XX**	**Response Required**	**Mandatory**	**99**
	$01	Response Required	Mandatory	99
	$02	No Response Required	Mandatory	99
2	**$XX**	**Event Window Time**	**Mandatory**	**104**
	$00	Reserved	N/A	
	$01	Tester Present Required	Optional	104
	$02	Infinite Time To Response	Optional	104
	$03 – $7F	Time interval (Resolution TBD)	Optional	104
	$80	No Event Window	Conditional[1]	104
	$81 - $FF	Reserved	N/A	
3	**$XX**	**Event Type**	**Mandatory**	**104**
	$00 - $7F	Reserved	N/A	
	$80	Report Activated Events	Optional	104
	$81	Stop Response On Event	Mandatory	104
	$82	On New DTC	Optional	104
	$83	On Timer Interrupt	Optional	104
	$84	On Change Of Record Value	Optional	105
	$85 - $9F	Reserved By Document	N/A	
	$A0	On Comparison of Values	Optional	105
	$A1- $AF	Event Type (ECU specific)	Optional	
	$B0 - $FF	Reserved By Document	N/A	
n	**$00 - FF**	**Event Type Parameter #1**	**Condition[2]**	
:	**:**		**:**	
	$00 -$FF	**Event Type Parameter #k**	**Condition[2]**	
n+k	**$XX**	**Service to Respond: Diagnostic Service ID**	**Mandatory**	
	$18	Read Diagnostic Trouble Code By Status	Optional	
	$21	Read Data By Local Identifier	Optional	
	$30	Input Output Control By Local Identifier	Optional	
	$31	Start Routine By Local Identifier	Optional	
	$32	Stop Routine By Local Identifier	Optional	
	$33	Request Routine Results By Local Identifier	Optional	
	$00-$FF	**Service to Respond: Diagnostic Service Parameter #1**	**Condition[3]**	
:		**:**	**:**	
M	**$00-$FF**	**Service to Respond: Diagnostic -Service Parameter #m**	**Condition[3]**	

Table 3.28.2-1 Response On Event Request Message Format

Condition[1]: Required if Event Type = Report Activated Events ($80) or Stop Response On Event($81) are supported

Condition[2]: Required if the Event Type requires additional parameters to be specified for the event to respond to.

Condition[3]: Required if the specified diagnostic service ID requires additional service parameters.

3.28.3 POSITIVE RESPONSE MESSAGE FORMAT (FOR ALL EVENT TYPES BUT "REPORT ACTIVATED EVENTS")

Data Byte #	Hex Value	Parameter Name	Message Usage	Reference Page
0	**$C6**	**Response On Event Request Service Positive Response ID**	**Mandatory**	
1	**$00 - $FF**	**Number Of Identified Events**	**Mandatory**	
2	**$XX**	**Event Window Time** The Event Window Time must be identical to the Event Window Time sent in the request message. Refer to **Table 3.28.2-1**	**Mandatory**	
3	**$XX**	**Activated Event #1: Event Type** The Event Type must be identical to the Event Type sent in the request message. Refer to **Table 3.28.2-1**	**Conditional**	
n	**$00 - FF**	**Activated Event #1: Event Type Parameter #1** Event Type Parameter #1 must be identical to Event Type Parameter #1 sent in the request message. Refer to **Table 3.28.2-1**	**Conditional**	
:	**:**	**:**		
n+m-1	**$00 - FF**	**Activated Event #1: Event Type Parameter #m** Event Type Parameter #m must be identical to Event Type Parameter #m sent in the request message. Refer to **Table 3.28.2-1**	**Conditional**	
	$XX	**Activated Event #1: Service to Respond: Diagnostic Service ID** The Diagnostic Service ID must be identical to Diagnostic Service ID sent in the request message. Refer to **Table 3.28.2-1**	**Mandatory**	
	$00-$FF	**Activated Event #1: Service to Respond: Diagnostic Service Parameter #1** The Diagnostic Service Parameter #1 must be identical to Diagnostic Service Parameter #1 sent in the request message. Refer to **Table 3.28.2-1**	**Conditional**	
:		**:**	**:**	
M	**$00-$FF**	**Activated Event #k: Service to Respond: Diagnostic -Service Parameter #m** The Diagnostic Service Parameter #m must be identical to Diagnostic Service Parameter #m sent in the request message. Refer to **Table 3.28.2-1**	**Conditional**	

Table 3.28.3-1 Response On Event Response Message Format

Condition: Conditional bytes shall be included in the Positive Response if specified in the Request Message of the activated Response On Event services.

3.28.4 POSITIVE RESPONSE MESSAGE FORMAT (EVENT TYPE: REPORT ACTIVATED EVENTS)

Data Byte #	Hex Value	Parameter Name	Message Usage	Reference Page
0	**$C6**	**Response On Event Request Service Positive Response ID**	**Mandatory**	
1	**$00 - $FF**	**Number Of Activated Events**	**Mandatory**	
2	**$80**	**No Event Window**	**Mandatory**	
3	**$XX**	**Event Type** The Event Type must be identical to the Event Type sent in the request message. Refer to **Table 3.28.2-1**	**Mandatory**	
n	**$00 - FF**	**Event Type Parameter #1** Event Type Parameter #1 must be identical to Event Type Parameter #1 sent in the request message. Refer to **Table 3.28.2-1**	**Conditional**	
:	**:**	**:**		
n	**$00 - FF**	**Event Type Parameter #m** Event Type Parameter #m must be identical to Event Type Parameter #m sent in the request message. Refer to **Table 3.28.2-1**	**Conditional**	
	$XX	**Service to Respond: Diagnostic Service ID** The Diagnostic Service ID must be identical to Diagnostic Service ID sent in the request message. Refer to **Table 3.28.2-1**	**Mandatory**	
	$00-$FF	**Service to Respond: Diagnostic Service Parameter #1** The Diagnostic Service Parameter #1 must be identical to Diagnostic Service Parameter #1 sent in the request message. Refer to **Table 3.28.2-1**	**Conditional**	
:		**:**	**:**	
M	**$00-$FF**	**Service to Respond: Diagnostic -Service Parameter #m** The Diagnostic Service Parameter #m must be identical to Diagnostic Service Parameter #m sent in the request message. Refer to **Table 3.28.2-1**	**Conditional**	
M+1 **:** **M+k**		**Repeat structure of bytes 3 to M according to Number Of Activated Events**		

Table 3.28.4-1 Response On Event Response Message Format

Condition: Conditional bytes shall be included in the Positive Response if specified in the Request Message.

3.28.5 NEGATIVE RESPONSE MESSAGE FORMAT

Data Byte #	Data Value	Parameter Description	Message Usage	Reference Page
0	**$7F**	**Negative Response**	**Conditional**	
1	**$86**	**Tester Present Request Service ID**	**Conditional**	
2	**$XX**	**Negative Response Code** Refer to Section 5 "**Negative Response Codes**" for global and diagnostic service specific negative response codes.	**Conditional**	

Table 3.28.5-1 Response On Event Negative Response Message

Condition: An ECU shall not return a **Negative Response ($7F)** if the **No Response Required ($02)** parameter was used by the diagnostic test tool in the request.

3.28.6 IMPLEMENTATION EXAMPLES OF RESPONSE ON EVENT

For the following examples, it is assumed that the Event Window Time parameter is calculated as follows:

$XX * 10 seconds = **Event Window Time** (e.g. $08 * 10 seconds = 80 seconds)

This is an example only and the **Event Window Time** may be calculated using any formula specified by the ECU Development Engineer except for certain values as specified in annex B2.

For these examples, the **Event Type** has the following encoding:

Data Value	Meaning	Number of Bytes in Event Type Parameters
$82	**On New DTC**	**1**
$84	**On Change Of Record Value**	**1**

EXAMPLE #1 – RESPONSE ON EVENT (FINITE EVENT WINDOW TIME)

Example #1 shows how to use the **Response On Event** service to get automatic updates for a specified set of data values. The condition is that another specified set of data has changed. The whole service is time limited.

For this example, the Event Type parameter for Event Type $84 (On Change Of Record Value) has the following encoding.

Data Value	Meaning
$09	Change in Local ID $A9, byte 5
$10	Change in Local ID $AA, byte 1
:	:
$20	Change in Local ID $AA, byte 10

Hex	Diagnostic Tool Request Message
$86	**Response On Event Request**
$01	Response Required = yes
$08	Event Window Time = 80 seconds
$84	Event Type = On Change Of Record Value
$20	Event Type Parameter = Change in Local ID $AA, byte 10
$21	Service to Respond: Service ID = Read Data By Local ID
$30	Service to Respond: Service ID Parameter = Record Local Identifier

Table 3.28.6-1 Implementation Example of Response On Event with Finite Event Window (Request)

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$C6	**Response On Event Request Positive Response**	**$7F**	**Negative Response Service Identifier**
$00	Number Of Identified Events	$86	Response On Event Request
$08	Event Window Time = 80 seconds	$XX	Negative Response Code
$84	Event Type = On Change Of Record Value		
$20	Event Type Parameter = Change in Local ID $AA, byte 10		
$21	Service to Respond: Service ID = Read Data By Local ID		
$30	Service to Respond: Service ID Parameter = Record Local Identifier		

Table 3.28.6-2 Implementation Example of Response On Event with Finite Event Window (Initial Response)

In this example, the duration between the ECU enabling the **Response On Event** service and the Local ID $AA, byte 10 changing is denoted as T1. (Note: T1 must be less than 80 seconds or the **Response On Event** service will timeout.)

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$61	**Read Data By Local Identifier Positive Response**	**$7F**	**Negative Response Service Identifier**
$30	Local Identifier	$86	Response On Event Request
$01	Record Value #1	$XX	Negative Response Code
$02	Record Value #2		
$03	Record Value #3		

Table 3.28.6-3 Implementation Example of Response On Event with Finite Event Window (Event Occurred)

Table 3.28.6-3 will repeat for each event. (i.e. each time the data is changed in Local ID $AA, byte 10 within the 80 second Event Window) For this example, the event occurs 3 times within the event window.

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$C6	**Response On Event Request Positive Response**	**$7F**	**Negative Response Service Identifier**
$03	Number Of Identified Events	$86	Response On Event Request
$08	Event Window Time = 80 seconds	$XX	Negative Response Code
$84	Event Type = On Change Of Record Value		
$20	Event Type Parameter = Change in Local ID $AA, byte 10		
$21	Service to Respond: Service ID = Read Data By Local ID		
$30	Service to Respond: Service ID Parameter = Record Local Identifier		

Table 3.28.6-4 Implementation Example of Response On Event with Finite Event Window (Final Response)

Note that Table 3.28.6-4 , byte 3 (Number of Identified Events) has incremented to $03 which corresponds to the number of times the event occurred during the 80 second event window.

EXAMPLE #2 – RESPONSE ON EVENT (INFINITE EVENT WINDOW)

Example #2 shows how to use the response On Event service to get the latest DTC every time a DTC is detected.

For this example, the Event Type parameter for Event Type $82 (On New DTC) has the following encoding

Data Value	Meaning
$01	Detected DTC
$02	Active DTC

Hex	Diagnostic Tool Request Message
$86	**Response On Event Request**
$01	Response Required = yes
$02	Event Window Time = Infinite
$82	Event Type = On New DTC
$01	Event Type Parameter = Detected DTC
$18	Service to Respond: Service ID = Read DTC By Status
$04	Service to Respond: Service ID Parameter = Request Most Recent DTC
$FF	Service to Respond: Service ID Parameter = (Group of DTC = All DTC's) MSB
$00	Service to Respond: Service ID Parameter = (Group of DTC = All DTC's) LSB

Table 3.28.6-5 Implementation Example of Response On Event with Infinite Event Window (Request)

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$C6	**Response On Event Request Positive Response**	**$7F**	**Negative Response Service Identifier**
$00	Number Of Identified Events	$86	Response On Event Request
$02	Event Window Time = Infinite	$XX	Negative Response Code
$82	Event Type = On New DTC		
$01	Event Type Parameter = Detected DTC		
$18	Service to Respond: Service ID = Read DTC By Status		
$04	Service to Respond: Service ID Parameter = On Most Recent DTC		
$FF	Service to Respond: Service ID Parameter = (Group of DTC = All DTC's) MSB		
$00	Service to Respond: Service ID Parameter = (Group of DTC = All DTC's) LSB		

Table 3.28.6-6 Implementation Example of Response On Event with Infinite Event Window (Initial Response)

Hex	ECU Positive Response Message	Hex	ECU Negative Response Message
$58	**Read DTC By Status Positive Response**	**$7F**	**Negative Response Service Identifier**
$01	Number Of DTC	$18	Read DTC By Status Positive Response
$01	DTC #1 [High Byte] {O2 Sensor Circuit Malfunction}	$XX	Negative Response Code
$30	DTC #1 [Low Byte] {Bank 1, Sensor 1}		
$24	Status Of DTC #1		

Table 3.28.6-7 Implementation Example of Response On Event with Infinite Event Window (Event Occurred)

Table 3.28.6-7 will repeat for each event. (i.e. each time a new error is detected). For this example, the ECU will continue returning DTC's until the request Response On Event with Event Type Stop Response On Event is sent or power down of the ECU.

4 DIAGNOSTIC SERVICE SUB FUNCTIONS AND PARAMETER OPTIONS

This section contains a list of parameters used by the Diagnostic Services presented in this specification and their associated options.

4.1 START DIAGNOSTIC SESSION, TESTER PRESENT

4.1.1 DIAGNOSTIC SESSION

Used by the **Start Diagnostic Session** service to select the specific behavior of an ECU.

NORMAL/DEFAULT SESSION ($81)

This diagnostic session enables the KWP2000 services used to support non-intrusive diagnostic functions. An ECU shall always start the Default Diagnostic Session when powered up. An ECU shall remain in Default Diagnostic Session if no other diagnostic session is started. No **Start Diagnostic Session Request ($10)** message with the **Diagnostic Session Normal/Default Session ($81)** is required after the connection has been made. From this **Diagnostic Session** it is possible to switch to any other diagnostic session.

In the default Diagnostic Session a specific set of diagnostic services is supported (see Table 3.2.1-1). In addition this **Diagnostic Session** enables all SAE J1979 / ISO 15031-5 related service ID's if the ECU is OBD II / EOBD compliant.

ECU FLASH REPROGRAMMING SESSION ($85)

This **Diagnostic Session** enables all Keyword Protocol 2000 services used to support the memory programming (Flash Reprogramming) of an ECU. This session will also be used in the development phase of the ECU. For a summary of all the service ID's that may be supported when an ECU is switched into **ECU Flash Reprogramming Session**, see Table 3.2.1-1 on page 8. (See Reference "*L"* in "Appendix A – *References*")
To keep this session active **Tester Present ($3E)** is required.

STAND BY SESSION ($89)

This Diagnostic Session enables the diagnostic tool to switch the ECU into a session of constant current consumption. The implementation of this diagnostic session applies to "slave" ECUs which control without manual operating actuators. (i.e. ECU's which react to temperature, pressure, self-tests, or requests from other ECU's.) Once an ECU has entered **Stand By Session**, if a request is made that could damage an ECU due to the constant level of current consumption, the request shall not be acted upon and the appropriate negative response shall be sent.

Note: ECU's which guarantee constant current consumption after "ignition on" without being in **Stand By Session**, do not need to support this session.

To keep this session active **Tester Present ($3E)** is required.

ECU PASSIVE SESSION ($90)

This **Diagnostic Session** enables the diagnostic tool to switch off the ECU's transmission of CAN-(or other network) messages for simulation purposes. By switching to a different diagnostic session or power down the transmission shall be re-enabled.
When the bus system transitions into sleep mode, an ECU in **ECU Passive Session ($10 $90)** shall also invoke sleep mode. When the bus system wakes-up, the ECU shall continue this session
In this session an ECU may remain active internally after disabling normal message transmission.
Note: this session shall only be used during the period of ECU development. **For production and service this functionality shall be removed from the ECU**. The services **Enable Normal Messages Transmission ($29)** and **Disable Normal Messages Transmission ($28)** shall be used instead.

EXTENDED DIAGNOSTIC SESSION ($92)

This **Diagnostic Session** enables all Keyword Protocol 2000 diagnostic services that are supported by DCX**.** **Extended Diagnostic Session** is an enhanced diagnostic session. For a summary of all the service ID's that should be supported if available when an ECU is switched into **Extended Diagnostic Session**, see Table 3.2.1-1 on page 8.
To keep this session active, any diagnostic request must be received by an ECU within P3max. If there are no diagnostic services being used to interrogate, modify, or alter an ECU, a **Tester Present ($3E)** is required.

4.2 ECU RESET

4.2.1 RESET MODE

Describes what type of reset an ECU has to perform.

POWER ON ($01)

This value identifies the **Power On Reset Mode** which shall be a simulated **Power On** reset which most ECU's perform after ignition OFF/ON cycle.

NONVOLATILE MEMORY RESET ($82)

Used to reset RAM only, not EEPROM

4.3 CLEAR DTC INFORMATION, READ STATUS OF DTC, AND READ DTC BY STATUS

4.3.1 GROUP OF DTC

Group Of DTC is used to specify a single DTC or a functional group of DTC's. When the **Group Of DTC** parameter is used with the **Read Diagnostic Trouble Codes By Status ($18)**, all DTC's associated with a specified functional group are returned. When the **Group Of DTC** parameter is used with **Clear Diagnostic Information ($14)**, all DTC's associated with a specified functional group or a single specified DTC shall be cleared.

Data Value	Parameter Description
$0000 - $3FFF	Powertrain DTC
$4000 - $7FFF	Chassis DTC
$8000 - $BFFF	Body DTC
$C000 - $FEFF	Network DTC

Table 4.3.1-1 Group Of DTC

ALL POWERTRAIN DTC'S ($0000)

This parameter option is used by:

- **Group Of DTC** in the **Read Diagnostic Trouble Codes By Status ($18)** service to return all Powertrain DTC's
- **Group Of DTC** in the **Clear Diagnostic Information ($14)** service to specify all Powertrain DTC's will be cleared.
- **Group Of DTC** in the **Control DTC Setting ($85)** service to specify that the setting of all Powertrain DTC's will be enabled or disabled.

ALL CHASSIS DTC'S ($4000)

- **Group Of DTC** in the **Read Diagnostic Trouble Codes By Status ($18)** service to return all Chassis DTC's
- **Group Of DTC** in the **Clear Diagnostic Information ($14)** service to specify all Chassis DTC's will be cleared.
- **Group Of DTC** in the **Control DTC Setting ($85)** service to specify that the setting of all Chassis DTC's will be enabled or disabled.

ALL BODY DTC'S ($8000)

- **Group Of DTC** in the **Read Diagnostic Trouble Codes By Status ($18)** service to return all Body DTC's
- **Group Of DTC** in the **Clear Diagnostic Information ($14)** service to specify all Body DTC's will be cleared.
- **Group Of DTC** in the **Control DTC Setting ($85)** service to specify that the setting of all Body DTC's will be enabled or disabled.

ALL NETWORK DTC'S ($C000)

- **Group Of DTC** in the **Read Diagnostic Trouble Codes By Status ($18)** service to return all Network DTC's
- **Group Of DTC** in the **Clear Diagnostic Information ($14)** service to specify all Network DTC's will be cleared.

- **Group Of DTC** in the **Control DTC Setting ($85)** service to specify that the setting of all Network DTC's will be enabled or disabled.

ALL DTC'S ($FF00)

- **Group Of DTC** in the **Read Diagnostic Trouble Codes By Status ($18)** service to return all All DTC's
- **Group Of DTC** in the **Clear Diagnostic Information ($14)** service to specify all All DTC's will be cleared.
- **Group Of DTC** in the **Control DTC Setting ($85)** service to specify that the setting of all All DTC's will be disabled.

4.3.2 DIAGNOSTIC TROUBLE CODE

DCA Only: For an overview of assigning values to DTC's, please refer to document "Diagnostic Trouble Code Requirements Definition". (See reference "K" in "Appendix A – *References*")

POWERTRAIN DTC ($0001 - $3FFF)

This parameter option is used by the **Read Status Of Diagnostic Trouble Codes ($17)** service to return a specific Powertrain DTC. This parameter option is used by **Group Of DTC** in the **Clear Diagnostic Information ($14)** service to specify a single Powertrain DTC will be cleared.

CHASSIS DTC ($4001 - $7FFF)

This parameter option is used by the **Read Status Of Diagnostic Trouble Codes ($17)** service to return a specific Chassis DTC. This parameter option is used by **Group Of DTC** in the **Clear Diagnostic Information ($14)** service to specify a single Chassis DTC will be cleared

BODY DTC ($8001 - $BFFF)

This parameter option is used by the **Read Status Of Diagnostic Trouble Codes ($17)** service to return a specific Body DTC. This parameter option is used by **Group Of DTC** in the **Clear Diagnostic Information ($14)** service to specify a single Body DTC will be cleared.

NETWORK DTC ($C001 - $FEFF)

This parameter option is used by the **Read Status Of Diagnostic Trouble Codes ($17)** service to return a specific Network DTC. This parameter option is used by **Group Of DTC** in the **Clear Diagnostic Information ($14)** service to specify a single Network DTC will be cleared.

4.3.3 NUMBER OF DTCS

Indicates how many **DTC's** specified in the **Group Of DTC** parameter are being reported by an ECU.

4.3.4 EXTENDED NUMBER OF DTCS

Indicates how many **DTC's** specified in the **Group Of DTC** parameter are being reported by an ECU. The Extended Number of DTCs is a 2 byte hex value.

4.3.5 STATUS OF DTC (AS USED BY THE DIAGNOSTIC TOOL REQUEST MESSAGE)

The **Status Of DTC,** used with the diagnostic tool request message, is used to specify the DTC's that are to be returned.

REQUEST IDENTIFIED DTC AND STATUS (SAE J2012 / ISO 15031-6 FORMAT) ($00)

This value is used by the diagnostic tool to indicate to the ECU that the ECU shall respond with all diagnostic trouble codes (DTC's) in the format defined by SAE J2012 / ISO 15031-6 which have been identified to cause a problem at the time of the request.

REQUEST SUPPORTED DTC AND STATUS (SAE J2012 / ISO 15031-6 FORMAT) ($01)

This value is used by the diagnostic tool to indicate to the ECU that the ECU shall respond with *all supported diagnostic trouble codes (DTC's) in the format defined by SAE J2012 / ISO 15031-6 regardless of their status*. Even if no DTC is stored the ECU shall report all DTC's which are supported. This allows the diagnostic tool to test the **DTC Readiness Flag** in the **Status Of DTC** parameter of each supported DTC.

REQUEST IDENTIFIED 2 BYTE HEX DTC AND STATUS ($02)

This value is used by the diagnostic tool to indicate to the ECU that the ECU shall respond with all diagnostic trouble codes (DTC's) in *2 byte hex* format which have been identified to cause a problem at the time of the request. The DTCs shall be reported by the ECU in the same sequence as they have been detected. Each DTC shall be reported by the ECU uniquely. (if an occurrence count is necessary it shall be reported in the corresponding environment / system supplier data of service $17.

This subfunction shall be supported for ECUs used for DCS

REQUEST SUPPORTED 2 BYTE HEX DTC AND STATUS ($03)

This value is used by the diagnostic tool to indicate to the ECU that the ECU shall respond with *all supported diagnostic trouble codes (DTC's) in 2 byte hex format regardless of their status*. Even if no DTC is stored the ECU shall report all DTC's which are supported. This allows the diagnostic tool to test the **DTC Readiness Flag** in the **Status Of DTC** parameter of each supported DTC.

REQUEST MOST RECENT 2 BYTE HEX DTC AND STATUS ($04)

This value is used by the diagnostic tool to indicate to the ECU that the ECU shall respond with the *most recently detected diagnostic trouble code (DTC) in 2 byte hex format and its status*. Only one DTC is reported.

REQUEST EXTENDED NUMBER OF SUPPORTED DTCS

This value is used by the diagnostic tool to indicate to the ECU that the ECU shall respond *with the total number of all supported diagnostic trouble codes (DTC's) in 2 byte hex format regardless of their status*.

4.3.6 STATUS OF DTC (AS USED BY THE ECU POSITIVE RESPONSE MESSAGE)

The **Status Of DTC,** used with the ECU positive response message, is used to report status information about each DTC, which has been identified at the time of the request, independent of their status. (i.e. stored, active, pending.) Table 4.3.6-1 shows the bit configuration for the **Status Of DTC** response.

Bit	Description of Status Of DTC-Response
3,2,1,0	Reserved. Set to all zero's. (i.e. 0000b)
4	**DTC Readiness Flag** This bit indicates whether the diagnostic trouble code test conditions have been met since the last successful **Clear Diagnostic Information** service. This bit allows for one bit to be available for service which indicates that the diagnostic test for a DTC has been completed, since DTC information was cleared. **'0' = "Test Complete for this DTC"** **'1' = "Test Not Complete for this DTC"** E.g. The **DTC Readiness Flag** of a kickdown switch of an Automatic Transmission Controller will only be set to the **Test Complete** status if the kickdown switch is activated (full throttle) and if all other DTC diagnostic test criteria have been met during the driving cycle. The other **Status Of DTC** parameter values shall only be updated by the software of the ECU if the **DTC Readiness Flag** is set to **Test Complete** or if the diagnostic tool has successfully performed a **Clear Diagnostic Information** service. After a **Clear Diagnostic Information** service, the **DTC Readiness Flag** shall be set to **Test Not Complete**.
6,5	**DTC Storage State** This bit field is state encoded and represents different diagnostic trouble code states. The state diagram of the **DTC Storage State** parameter in Figure 5 shall provide more detail about the possible **DTC Storage State** conditions in the ECU's memory. **'00' = "No DTC Detected at time of request";** no DTC is stored in non volatile memory; **'01' = "DTC Not Present at time of request (Stored DTC)";** DTC is no longer present; DTC is stored in non volatile memory; **'10' = "DTC Maturing-Pending at time of request"** DTC has not met maturing criteria to become an active/stored DTC but failure has been detected; **'11' = "DTC Present at time of request (Active DTC)";** DTC is present; DTC is stored in non volatile memory;
7	**Warning Indicator Request State** This bit shall report the status of any warning indicators associated with a particular DTC. Warning outputs may consist of indicator lamp(s), displayed text information, etc. If no warning indicators exist for a given system or particular DTC, this status shall default to a logic "0" state. If the warning output is on for a given DTC, the stored status flag shall also be true. Reset to a logical '0' after a call to Clear Diagnostic Information. Additional reset conditions defined by vehicle manufacturer / implementation. 1 = Warning indicator requested to be ON. 0 = Warning indicator not requested to be ON

Table 4.3.6-1 Definition of Status Of DTC values

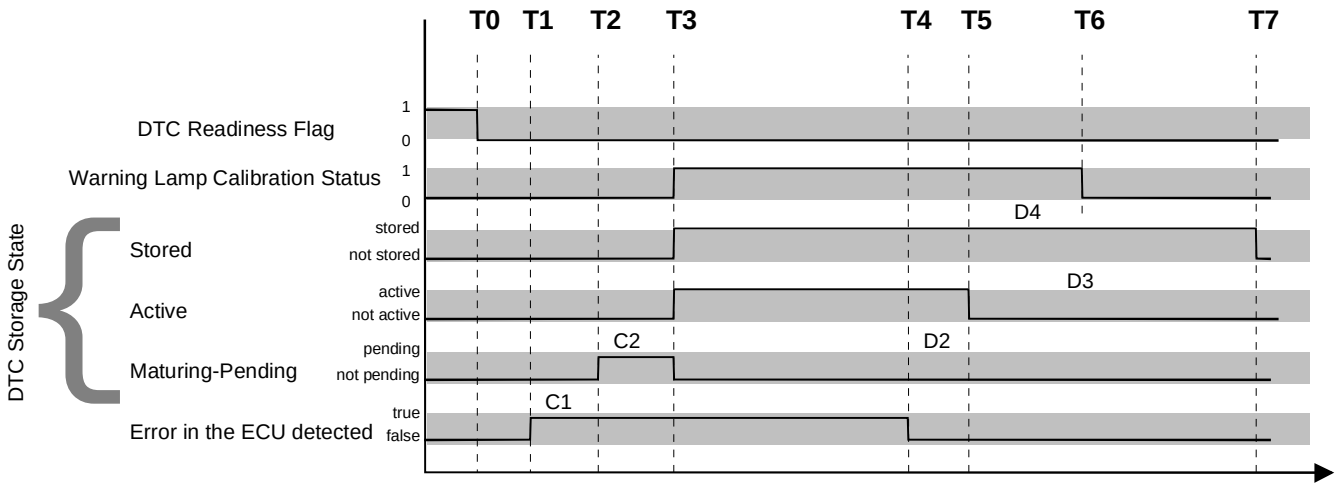

Figure 5 Diagram of Status of DTC

The example in Figure 5 illustrates how the bits 4 to 7 in the Status of DTC are being used. At T0 all conditions to execute the error check function of a specific DTC of the ECU have been met and the error check function has been run. If the function has been run successfully at least once, the DTC Readiness Flag is set to 0. If the error check function result is negative, the DTC readiness flag shall only be set to 0 after all conditions have been met to verify the error. (i.e. If a fault is detected the DTC readiness flag is set to 0 at the same time the DTC storage state becomes active.) At T1 the error check function detects this fault. After condition C1 is met (T2), the DTC Storage State is set to maturing-pending. After condition C2 is met (T3), the fault becomes active and stored. If the error requires the warning lamp on, the warning indicator request state bit is set to 1. After the error is no longer detected (T4), the bits are set to 0 according to their respective de-maturing condition. (D2 to D4)

Note 1: Conditions
A condition could be a specified time or for example 2 driving cycles; there need not be a condition for every transition to a different state. The condition could also be “immediately” or “never”.

Note 2: For an OBDII / EOBD compliant ECU:
The storage state 'pending' will accommodate the OBD II / EOBD requirements for a “one malfunction” condition of a two-trip failure.

4.3.7 SYSTEM SUPPLIER DATA
Any data associated with a **DTC** including, but not limited to, environmental data and freeze frame data.

4.4 READ ECU IDENTIFICATION, READ DATA BY LOCAL IDENTIFIER, DYNAMICALLY DEFINE DATA BY LOCAL IDENTIFIER, WRITE DATA BY LOCAL IDENTIFIER, READ DATA BY IDENTIFIER, WRITE DATA BY IDENTIFIER

4.4.1 ECU IDENTIFICATION PARAMETER
This parameter is used in the **Read ECU Identification Positive Response ($5A)** message to return data pertaining to the **Identification Option**. For example, if **Identification Option** = **VIN ($90)**, then the **ECU Identification** parameters would return the **VIN**.

4.4.2 IDENTIFIER
Identifiers are 2 byte numeric logical abstractions of a data element residing in the ECU. Logical addressing of parameters promotes global references of data elements thereby allowing ID’s to retain their original defined functions independent of changes to actual physical memory addresses. Whereas diagnostic services pertain to general functional requests for data, ID’s are used to uniquely identify data elements in the ECU (i.e. sensor data, fault data, configuration, software variables, etc.). ID’s commonly point to memory locations in RAM, ROM, or EEPROM, and are logically grouped together by type to permit further expansion without compromising LID groupings.

4.4.3 LOCAL IDENTIFIER

Local Identifiers are 1 byte numeric logical abstractions of a data element residing in the ECU. Logical addressing of parameters promotes global references of data elements thereby allowing LID's to retain their original defined functions independent of changes to actual physical memory addresses. Whereas diagnostic services/test modes pertain to general functional requests for data, LID's are used to uniquely identify data elements in the ECU (e.g sensor data, fault data, configuration, software variables, etc.). LID's commonly point to memory locations in RAM, ROM, or EEPROM, and are logically grouped together by type to permit further expansion without compromising LID groupings. If the **Local Identifier** is dynamically defined by the **Dynamically Define Local Identifier ($2C)** then the data should be temporarily stored in RAM. These local identifiers are completely independent of the local identifiers used by the **Input Output Control By Local Identifier Request ($30)** service.

4.4.4 IDENTIFICATION OPTION

Provides the identification data to the diagnostic tool via the **Read ECU Identification Positive Response ($5A)**.

DCS ECU IDENTIFICATION ($86)

Upon requesting $86 with the diagnostic service **Read ECU Identification ($1A)**, the ECU shall return the ECU Identification which is used by the tester to uniquely identify the ECU. The structure of the positive response message shall follow Table 3.7.3-1 and the content of the local identifier will match *Table 4.4.4-1 DCS ECU Identification Message Structure*. The format of the DCS part number in bytes 2 to 6 shall conform to DCS corporate specification.
Note: for the partnumber 123 456 78 90 "12" is the high byte.

Byte	Title	Description
2	**Number Positions 10 | 9 (High Byte)**	Bytes 0 to 4 are used for the Part Number. The format shall be BCD.
3	**Number Position 8 | 7**	Bytes 0 to 4 are used for the Part Number. The format shall be BCD.
4	**Number Positions 6 | 5**	Bytes 0 to 4 are used for the Part Number. The format shall be BCD.
5	**Number Positions 4 | 3**	Bytes 0 to 4 are used for the Part Number. The format shall be BCD.
6	**Number Positions 2 | 1 (Low Byte)**	Bytes 0 to 4 are used for the Part Number. The format shall be BCD.
7	**ECU Hardware Build Date (WW)**	This byte used in conjunction with byte 9 to return the date the ECU hardware was built. This byte is used to return the week the ECU hardware was built. Format is BCD.
8	**ECU Hardware Build Date (YY)**	This byte used in conjunction with byte 10 to return the date the ECU hardware was built. This byte is used to return the year the ECU hardware was built. Format is BCD.
9	**ECU Software Written Date (WW)**	This byte used in conjunction with byte 7 to return the date the ECU software was written. This byte is used to return the week the ECU software was written. Format is BCD.
10	**ECU Software Written Date (YY)**	This byte used in conjunction with byte 8 to return the date the ECU software was written. This byte is used to return the year the ECU software was written. Format is BCD.
11	**Supplier Identification**	This byte shall cause the ECU to return the supplier identification. The values assigned to the supplier identification number are defined in the Supplier Identification Reference (See reference "*N*" in "Appendix A – **References"**)
12	**Diagnostic Information (High Byte)**	This byte used in conjunction with Diagnostic Information (Low Byte) causes the ECU to return diagnostic information Bit 7: 0 = in series production; 1 = in development Bits 6 to 0: ECU type ID allocated by DCX Diagnostics
13	**Diagnostic Information (Low Byte)**	This byte used in conjunction with Diagnostic Information (High Byte) causes the ECU to return diagnostic information Bit 7 to 0: Diagnostic Version – The diagnostics version begins with a starting value and is incremented with every diagnostics-relevant(i.e. DIOGENES relevant) change to the software (function software, bootsoftware). The following holds true for the function software: Value Range: $00 - $DF Start Value: $00 (For both development and series production) The following holds true the boot software: Value Range: $E0 - $FF Start Value: $E0 (For both development and series production)
14	**Reserved for Future Definition by DCX**	This is only a filler and currently only $00 should be put in this position as a place holder.
15	**ECU Production Date (YY)**	This byte used in conjunction with bytes 1 and 0 to return the production date of the ECU. This byte is used to return the production year of the ECU . The format shall be BCD.
16	**ECU Production Date (MM)**	This byte used in conjunction with bytes 2 and 0 to return the production date of the ECU. This byte is used to return the production month of the ECU. The format shall be BCD.
17	**ECU Production Date (DD)**	This byte used in conjunction with bytes 1 and 0 to return the production date of the ECU. This byte is used to return the production day of the ECU. The format shall be BCD

Table 4.4.4-1 DCS ECU Identification Message Structure

Note: The data corresponding to local identifier $86 begins with byte 2 as byte 0 is reserved for the diagnostic service and byte 1 is reserved for the local identifier.

DCX / MMC ECU IDENTIFICATION ($87)

Upon requesting $87 with the diagnostic service **Read ECU Identification ($1A)**, the ECU shall return the ECU Identification which is used by the tester to uniquely identify the ECU. The structure of the positive response message shall follow Table 3.7.3-1 and the content of the local identifier will match Table 4.4.4-2.

Byte	Title	Encoding	Description
2	ECU Origin	HEX	Please refer to "See reference Q in "Appendix A – **References**"
3	Supplier Identification	HEX	The values assigned to the supplier identification number are defined in the Supplier Identification Reference (See reference "*N*" in "Appendix A – **References**")
4	Diagnostic Information (High Byte)	HEX	**Variant** These bits are used to distinguish ECU's that have the same diagnostic CAN identifiers and the same diagnostic protocol (KW2000) and the same supplier code (byte 2). **DCS** Bit 7: 0 - indicates production status 1 - indicates development status Bit 6-0: ECU Identification **DCA** Bit 7: 0 – Does not support LID $EA 1 – Does Support LID $EA Bit 6-0: ECU Identification (Refer to ECU specific DDT for more information.)
5	Diagnostic Information (Low Byte)	HEX	**Diagnostic Version** The diagnostic version begins with a starting value and is incremented with every diagnostics relevant (i.e. diagnostic tool / DIOGENES relevant) changes to the software in the ECU (Functional SW or Boot SW) The following values are reserved for the Functional SW (Code) Value range: $00 - $DF Starting Value: $00 The following values are reserved for the Boot SW Value range: $E0 - $FF Starting Value: $E0 **DCA**: ECUs may also use this to specify both the Diagnostic Level and version number. (e.g. Level 2, version 4 = $24)
6	Reserved	HEX	**Reserved for future use.** Set to $FF
7	Hardware Version (Major Byte)	BCD	**Major Version**
8	Hardware Version (Minor Byte)	BCD	**Minor Version**
9	Software Version (High Byte)	BCD	**Major Version** (e.g. Software Version XX YY ZZ, this is XX)
10	Software Version (Middle Byte)	BCD	**Middle Version** (e.g. Software Version XX YY ZZ, this is YY)
11	Software Version (Low Byte)	BCD	**Minor Version** (e.g. Software Version XX YY ZZ, this is ZZ)
12	Part Number Position #1 (High Byte)	ASCII	DCA: The format of the part number in bytes 12 to 21 shall conform to the corporate specification: "Part Management Process Standard CEP-008A". DCS: The format of the DCS part number in bytes 12 to 21 shall conform to corporate specification.
13	Part Number Position #2	ASCII	
14	Part Number Position #3	ASCII	
15	Part Number Position #4	ASCII	
16	Part Number Position #5	ASCII	
17	Part Number Position #6	ASCII	
18	Part Number Position #7	ASCII	
19	Part Number Position #8	ASCII	
20	Part Number Position #9	ASCII	
21	Part Number Position #10 (Low Byte)	ASCII	

Table 4.4.4-2 DCX ECU Identification Message Structure

Note: The data corresponding to local identifier $87 begins with byte 2 as byte 0 is reserved for the diagnostic service and byte 1 is reserved for the local identifier.

VEHICLE IDENTIFICATION NUMBER(VIN) – ORIGINAL ($88)

Upon requesting $88 with the diagnostic service **Read ECU Identification ($1A)**, the ECU shall return the original Vehicle Identification Number. The structure of the positive response message shall follow Table 3.7.3-1 and the content of the local identifier will match Table 4.4.2-3. The format of the VIN shall conform to corporate specification: "VIN Coding Guide". (See Reference "*O"* in "Appendix A – *References*")

Byte	Title	Encoding	Description
2	VIN (MSB)	ASCII	
:	:		
18	VIN (LSB)	ASCII	

Table 4.4.4-3 Vehicle Identification Number (VIN) – Original and Current

Note: The data corresponding to local identifier $88 begins with byte 2 as byte 0 is reserved for the diagnostic service and byte 1 is reserved for the local identifier.

DIAGNOSTIC VARIANT CODE ($89)

Upon requesting $89 with the diagnostic service **Read ECU Identification ($1A)**, the ECU shall return the Diagnostic Variant Code which is used by the tester to uniquely identify the Diagnostic Variant. The structure of the positive response message shall follow Table 3.7.3-1 and the content of the local identifier will match Table 4.4.4-4

Note for implementation at MMC: The Diagnostic Variant Code shall only be used so that MMC's tester (MUT) can distinguish variations of diagnostic content implemented in ECU. The value of this code must be coordinated with MMC.

Byte	Title	Encoding	Description
2	Diagnostic Variant Code (High Byte)	HEX	
3	Diagnostic Variant Code	HEX	
4	Diagnostic Variant Code	HEX	
5	Diagnostic Variant Code (Low Byte)	HEX	

Table 4.4.4-4 Diagnostic Variant Code Message Structure

Note: The data corresponding to local identifier $89 begins with byte 2 as byte 0 is reserved for the diagnostic service and byte 1 is reserved for the local identifier.

VEHICLE IDENTIFICATION NUMBER(VIN) – CURRENT ($90)

Upon requesting $90 with the diagnostic service **Read ECU Identification ($1A)**, the ECU shall return the current Vehicle Identification Number. The structure of the positive response message shall follow Table 3.7.3-1 and the content of the local identifier will match Table 4.4.4-3 with the exception of using a $90 instead of an $88 as the local identifier. The format of the VIN shall conform to corporate specification: "VIN Coding Guide".(See Reference "*O"* in "Appendix A – *References*")

CALIBRATION ID ($96) (= SAEJ1979 / ISO 15031-5 OBD $09 $04)

Upon requesting $96 with the diagnostic service **Read ECU Identification ($1A)**, the ECU shall return the calibration ID. The following excerpt from ISO 15031-5 defines Calibration ID as:
"Multiple calibration identifications may be reported for a controller, depending on the software architecture. Calibration identifications can include a maximum of sixteen (16) characters. Each calibration identification can contain only printable ASCII characters, and will be reported as ASCII values. Any unused data bytes shall be reported as $00 and filled at the end of the calibration identification.
Calibration identifications shall uniquely identify the software installed in the ECU. There are currently proposed US OBD / European EOBD regulations that will require calibration identifications to be reported for emission-related software in a standardised format.
Calibrations developed by any entity other than the vehicle manufacturer shall also contain unique calibration identification to indicate that a calibration is installed in the vehicle that is different from that developed by the vehicle manufacturer.
Vehicle controllers that contain calibration identifications shall store and report sixteen (16) ASCII character calibration identifications, even though they may not use all sixteen (16) characters. This will allow modified calibration IDs to be reported that include additional characters."
The structure of the positive response message shall follow Table 3.8.3-1 and the content of the local identifier will match Table 4.4.4-54.

Byte	Title	Encoding	Description
2	Calibration ID (MSB)	ASCII	
:		:	
17	Calibration ID (LSB)	ASCII	

Table 4.4.4-5 Calibration ID

Note: The data corresponding to local identifier $96 begins with byte 2 as byte 0 is reserved for the diagnostic service and byte 1 is reserved for the local identifier.

CALIBRATION VERIFICATION NUMBER ($97) (= SAEJ1979 / ISO 15031-5 OBD $09 $06)

Upon requesting $97 with the diagnostic service **Read ECU Identification ($1A)**, the ECU shall return the Calibration Verification Number (CVN). The structure of the positive response message shall follow Table 3.8.3-1 and the content of the local identifier will match Table 4.4.4-6. The CVN is dynamically calculated by the ECU using a unique security hashing function.

Byte	Title	Encoding	Description
2	CVN Byte #1	HEX	
3	CVN Byte #2	HEX	
4	CVN Byte #3	HEX	
5	CVN Byte #4	HEX	

Table 4.4.4-6 Calibration Verification Identification

Note: The data corresponding to local identifier $97 begins with byte 2 as byte 0 is reserved for the diagnostic service and byte 1 is reserved for the local identifier.

ECU CODE FINGERPRINT ($9A)

Upon requesting $9A with the diagnostic service **Read ECU Identification ($1A)**, the ECU shall return the code programming date and tester serial number information. The structure of the positive response message shall follow Table 3.7.3-1 and the content of the local identifier will match Table 4.4.4-7.
Note: For more than one module, repeat bytes 4 to 11 according to the number of modules.

Byte	Title	Encoding	Description
2	# of Modules m	HEX	Identifies the number of modules included in the ECU.
3	Active Logical Block	HEX	Inditifies the Logical Block to be erased (for independent erase routines). $00 – No erase performed $01 – Erase 1st logical block $02 – Erase 2nd logical block : $FD – Erase the 254th logical block $FE – Erase all memory $FF – Reserved
4	#1 Tool Supplier Identification	HEX	If Tool Suplier does not match those specified in **Supplier Identification Reference**, set to $FF.
5	#1 Programming Date (MSB)	BCD	Year (YY)
6	#1 Programming Date	BCD	Month (MM)
7	#1 Programming Date (LSB)	BCD	Day (DD)
8	#1 Tester Serial Number	HEX	If byte 3 was set to $FF, set this byte to $FF
9	#1 Tester Serial Number	HEX	If byte 3 was set to $FF, set this byte to $FF
10	#1 Tester Serial Number	HEX	If byte 3 was set to $FF, set this byte to $FF
11	#1 Tester Serial Number	HEX	If byte 3 was set to $FF, set this byte to $FF
...			
N-7	#m Tool Supplier Identification	HEX	If Tool Suplier does not match those specified in **Supplier Identification Reference**, set to $FF.
...			
N	#m Tester Serial Number	HEX	If byte N-7 was set to $FF, set this byte to $FF

Table 4.4.4-7 ECU Code Fingerprint ($9A), ECU Data Fingerprint ($9B), and ECU Boot Fingerprint ($9F)

Note: The data corresponding to local identifiers $9A, $9B, and $9F begin with byte 2 as byte 0 is reserved for the diagnostic service and byte 1 is reserved for the local identifier.

ECU DATA FINGERPRINT ($9B)

Upon requesting $9B with the diagnostic service **Read ECU Identification ($1A)**, the ECU shall return the data programming date and tester serial number information. The structure of the positive response message shall follow Table 3.7.3-1 and the content of the local identifier will match Table 4.4.4-7 with the exception of using a $9B instead of a $9A as the local identifier.

Note: Byte 4 to 11 shall be repeated according to the number of modules.

ECU CODE SOFTWARE IDENTIFICATION ($9C)

Upon requesting $9C with the diagnostic service **Read ECU Identification ($1A)**, the ECU shall return the application code software information. The structure of the positive response message shall follow Table 3.7.3-1 and the content of the local identifier will match Table 4.4.4-8

Note: For more than one module, repeat bytes 4 to 20 according to the number of modules.

Byte	Title	Encoding	Description
2	# of Modules m	HEX	Identifies the number of modules included in the ECU.
3	ECU Origin	HEX	Please refer to Appendix A, Reference Q.
4	#1 Supplier Identification	HEX	The values assigned to the supplier identification number are defined in the Supplier Identification Reference (See reference "N" in "Appendix A – *References*")
5	#1 Diagnostic Information (High Byte)	HEX	**Variant Byte** These bits are used to distinguish ECU's that have the same diagnostic CAN identifiers and the same diagnostic protocol (KW2000) and the same supplier code (byte 2). **DCS** Bit 7: 0 - indicates production status 1 - indicates development status Bit 6-0: ECU Identification **DCA** Bit 7: 0 – Does not support LID $EA 1 – Does Support LID $EA Bit 6-0: ECU Identification (Refer to ECU specific DDT for more information.)
6	#1 Diagnostic Information (Low Byte)	HEX	**Diagnostic Version** The diagnostic version begins with a starting value and is incremented with every diagnostics relevant (i.e. DIOGENES relevant) changes to the software in the ECU (Functional SW or Boot SW) Note: For DCA, the value should equal the LevXX specified in the file name for the Diagnostic Definition Table. (e.g. For 04_ECU_Vehicle_Lev20.xls, the Diagnostic Version=$20) The following values are reserved for the Functional SW (Code) Value range: $00 - $DF Starting Value: $00 The following values are reserved for the Boot SW Value range: $E0 - $FF Starting Value: $E0
7	#1 (Reserved)	HEX	Set to $FF

8	#1 Software Version (High Byte)	BCD	**Major Version** (e.g. Software Version XX YY ZZ, this is XX)
9	#1 Software Version (Middle Byte)	BCD	**Middle Version** (e.g. Software Version XX YY ZZ, this is YY)
10	#1 Software Version (Low Byte)	BCD	**Minor Version** (e.g. Software Version XX YY ZZ, this is ZZ)
11	#1 Part Number Position #1 (High Byte)	ASCII	DCA: The format of the part number in bytes 11 to 20 shall conform to the corporate specification: "Part Management Process Standard CEP-008A". DCS: The format of the DCS part number in bytes 11 to 20 shall conform to corporate specification.
12	#1 Part Number Position #2	ASCII	
13	#1 Part Number Position #3	ASCII	
14	#1 Part Number Position #4	ASCII	
15	#1 Part Number Position #5	ASCII	
16	#1 Part Number Position #6	ASCII	
17	#1 Part Number Position #7	ASCII	
18	#1 Part Number Position #8	ASCII	
19	#1 Part Number Position #9	ASCII	
20	#1 Part Number Position #10 (Low Byte)	ASCII	
...			
N-16	#m Supplier Identification	HEX	The values assigned to the supplier identification number are defined in the Supplier Identification Reference (See reference "N" in "Appendix A – *References*")
...			
N	#m Part Number Position #10 (Low Byte)	ASCII	

Table 4.4.4-8 ECU Code Software Identification ($9C), ECU Data Software Identification ($9D), and ECU Boot Software Identification ($9E)

Note: The data corresponding to local identifiers $9C, $9D, and $9E begin with byte 2 as byte 0 is reserved for the diagnostic service and byte 1 is reserved for the local identifier.

ECU DATA SOFTWARE IDENTIFICATION ($9D)

Upon requesting $9D with the diagnostic service **Read ECU Identification ($1A)**, the ECU shall return the application code data information. The structure of the positive response message shall follow Table 3.7.3-1 and the content of the local identifier will match Table 4.4.4-8 with the exception of using a $9D instead of a $9C as the local identifier.

Note: For m additional modules, repeat bytes 4 to 20 m times.

ECU BOOT SOFTWARE IDENTIFICATION ($9E)

Upon requesting $9E with the diagnostic service **Read ECU Identification ($1A)**, the ECU shall return bootloader information. The structure of the positive response message shall follow Table 3.7.3-1 and the content of the local identifier will match Table 4.4.4-8 with the exception of using a $9E instead of a $9C as the local identifier.
Note: For m additional modules, repeat bytes 4 to 20 m times.

ECU BOOT FINGERPRINT ($9F)

Upon requesting $9F with the diagnostic service **Read ECU Identification ($1A)**, the ECU shall return the boot programming fingerprint data. The structure of the positive response message shall follow Table 3.7.3-1 and the content of the local identifier will match Table 4.4.4-7 with the exception of using a $9F instead of a $9A as the local identifier.

Note: For m additional modules, repeat bytes 4 to 11 m times.

DEVELOPMENT DATA ($E0)

Upon requesting $E0 with the diagnostic service **Read Data By Local Identifier ($21)**, the ECU shall return development data. The structure of the positive response message shall follow Table 3.8.3-1 and the content of the local identifier will match Table 4.4.4-9.

Byte	Title	Encoding	Description
2	Processor Type (MSB)	HEX	Please refer to Appendix A, Reference Q.
3	Processor Type (LSB)	HEX	

4	Communication Matrix Version (MSB)	BCD	Calender Week (WW)
5	Communication Matrix Version (LSB)	BCD	Year (YY)
6	CAN Driver Version (MSB)	BCD	For CAN Driver Version XX.YY, the MSB shall be XX and the LSB shall be YY. For example: 3.14: 03=MSB, 14=LSB.
7	CAN Driver Version (LSB)	BCD	
8	NM Version (MSB)	BCD	For NM Version XX.YY, the MSB shall be XX and the LSB shall be YY. For example: 3.14: 03=MSB, 14=LSB.
9	NM Version (LSB)	BCD	
10	KWP 2000 Module Version (MSB)	BCD	For KWP2000 Module Version XX.YY, the MSB shall be XX and the LSB shall be YY. For example: 3.14: 03=MSB, 14=LSB.
11	KWP 2000 Module Version (LSB)	BCD	
12	Transport Layer Version (MSB)	BCD	For Transport Layer Version XX.YY, the MSB shall be XX and the LSB shall be YY. For example: 3.14: 03=MSB, 14=LSB.
13	Transport Layer Version (LSB)	BCD	
14	DBKOM Version (MSB)	BCD	For DBKom Version XX.YY, the MSB shall be XX and the LSB shall be YY. For example: 3.14: 03=MSB, 14=LSB.
15	DBKOM Version (LSB)	BCD	
16	Flexer Version (MSB)	BCD	For Flexer Version XX.YY, the MSB shall be XX and the LSB shall be YY. For example: 3.14: 03=MSB, 14=LSB. Note for DCA: Flexer is currently unsupported. Please use 9999.
17	Flexer Version (LSB)	BCD	
18	Reserved	BCD	
19	Reserved	BCD	

Table 4.4.4-9 Development Data

Note: The data corresponding to local identifier $E0 begins with byte 2 as byte 0 is reserved for the diagnostic service and byte 1 is reserved for the local identifier.

ECU SERIAL NUMBER ($E1)

Upon requesting $E1 with the diagnostic service **Read Data By Local Identifier ($21)**, the ECU shall return the ECU serial number. The structure of the positive response message shall follow Table 3.8.3-1 and the content of the local identifier will match Table 4.4.4-10.

Byte	Title	Encoding	Description
2	ECU Serial Number (MSB)	See Description	DCA: The format of the ECU serial number shall conform to the corporate specification: "Component Parts Trace-ability Process Standard PS-10125" and shall be encoded in ASCII format. DCS: The format of the ECU serial number shall conform to the model line related specification.
:	:	:	
n	ECU Serial Number (LSB)	See Description	

Table 4.4.4-10 ECU Serial Number

Note: The data corresponding to local identifier $E1 begins with byte 2 as byte 0 is reserved for the diagnostic service and byte 1 is reserved for the local identifier.

DBCOM DATA ($E2)

Upon requesting $E2 with the diagnostic service **Read Data By Local Identifier ($21)**, the ECU shall return DBCom data. This information is being used for upload and download of the DBCom parameters (communication matrix). The structure of the positive response message shall follow Table 3.8.3-1 and the content of the local identifier will match Table 4.4.4-11.

Byte	Title	Encoding	Description
2	Memory Address Flash (High Byte)	HEX	
3	Memory Address Flash (Middle Byte)	HEX	
4	Memory Address Flash (Low Byte)	HEX	
5	Data Format Identifier Flash	HEX	
6	uncompressed Memory Size Flash (High Byte)	HEX	
7	uncompressed Memory Size Flash (Middle Byte)	HEX	
8	uncompressed Memory Size Flash (Low Byte)	HEX	
9	Memory Address RAM (High Byte)	HEX	
10	Memory Address RAM (Middle Byte)	HEX	
11	Memory Address RAM (Low Byte)	HEX	
12	uncompressed Memory Size RAM (High Byte)	HEX	
13	uncompressed Memory Size RAM (Middle Byte)	HEX	
14	uncompressed Memory Size RAM (Low Byte)	HEX	
15	Memory Address EEPROM (High Byte)	HEX	
16	Memory Address EEPROM (Middle Byte)	HEX	
17	Memory Address EEPROM (Low Byte)	HEX	
18	uncompressed Memory Size EEPROM (High Byte)	HEX	
19	uncompressed Memory Size EEPROM (Middle Byte)	HEX	
20	uncompressed Memory Size EEPROM (Low Byte)	HEX	

Table 4.4.4-11 DBCom Data

Note: The data corresponding to local identifier $E2 begins with byte 2 as byte 0 is reserved for the diagnostic service and byte 1 is reserved for the local identifier.

OPERATING SYSTEM VERSION ($E3)

Upon requesting $E3 with the diagnostic service **Read Data By Local Identifier ($21)**, the ECU shall return the version of the operating system used by the ECU. The structure of the positive response message shall follow Table 3.8.3-1 and the content of the local identifier will match Table 4.4.4-12.

Byte	Title	Encoding	Description
2	Operating System (MSB)	HEX	
:		HEX	
n	Operating System (LSB)	HEX	

Table 4.4.4-12 Operating System Version

Note: The data corresponding to local identifier $E3 begins with byte 2 as byte 0 is reserved for the diagnostic service and byte 1 is reserved for the local identifier.

REPROGRAMMING FAULT REPORTING ($E4)

The contents and usage requirements of this Local ID shall be found in the ECU Flash Reprogramming Requirements Definition (See Reference “L” in “Appendix A – *References*”)

VEHICLE INFORMATION ($E5)

Upon requesting $E5 with the diagnostic service **Read Data By Local Identifier ($21)**, the ECU shall return the model year, vehicle, body style, and country code. The structure of the positive response message shall follow Table 3.8.3-1 and the content of the local identifier will match Table 4.4.4-13.

Byte	Title	Encoding	Description
2	Model Year	HEX	Please refer to the ECU specific DDT.
3	Vehicle	HEX	
4	Body Style	HEX	
5	Country Code	HEX	

Table 4.4.4-13 Vehicle Information

Note: The data corresponding to local identifier $E5 begins with byte 2 as byte 0 is reserved for the diagnostic service and byte 1 is reserved for the local identifier.

FLASH INFO 1 ($E6)

Upon requesting $E6 with the diagnostic service **Read Data By Local Identifier ($21)**, the ECU shall return error information of a SID in case of negative response. This information is generated by the flash reprogramming software module and is used during development. The structure of the positive response message shall follow Table 3.8.3-1 and the content of the local identifier will match Table 4.4.4-14. (See Reference "*M*" in "Appendix A – *References*")

Byte	Title	Encoding	Description
2	Flash Info 1 (byte 1)	HEX	
:			
n	Flash Info 1 (byte n)	HEX	

Table 4.4.4-14 Flash Info 1

Note: The data corresponding to local identifier $E6 begins with byte 2 as byte 0 is reserved for the diagnostic service and byte 1 is reserved for the local identifier.

FLASH INFO 2 ($E7)

Upon requesting $E7 with the diagnostic service **Read Data By Local Identifier ($21)**, the ECU shall return 19 bytes of information of hardware scanning as entry function prior to a flash reprogramming procedure, and to get statistical information after flash reprogramming procedure. This information is generated by the flash reprogramming software module and is used during development. The structure of the positive response message shall follow Table 3.8.3-1 and the content of the local identifier will match Table 4.4.4-15. (See Reference "*M*" in "Appendix A – *References*")

Byte	Title	Encoding	Description
2	Flash Info 2 (byte 1)	HEX	
:		:	
n	Flash Info 2 (byte n)	HEX	

Table 4.4.4-15 Flash Info 2

Note: The data corresponding to local identifier $E7 begins with byte 2 as byte 0 is reserved for the diagnostic service and byte 1 is reserved for the local identifier.

SYSTEM DIAGNOSTIC GENERAL PARAMETER DATA ($E8)

Upon requesting $E8 with the diagnostic service **Read Data By Local Identifier ($21)**, the ECU shall return general configuration data of the SDCOM-SW-module. In Byte 1 the ECU transmits the internal communication mode and the information if global process data exist. The coding of Byte 1 is shown in Figure 6.

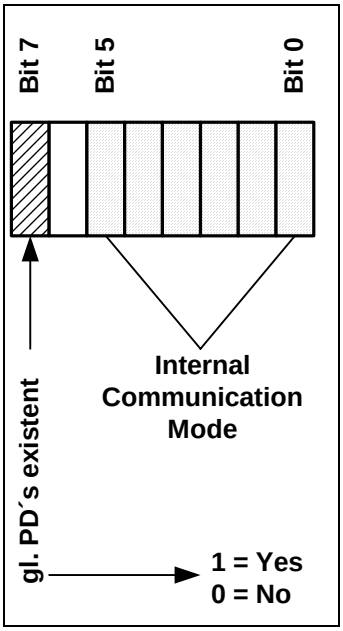

Figure 6 Coding Byte 1 – System diagnostic General Parameter Data

Byte	Title	Encoding	Description
2	Internal Communication Mode	See Above	Internal Communication Mode and Information Global Process Data Existent
3	SDCOM Main Version	BCD	SDCOM-SW-Modul Versioninformation
4	SDCOM Inter Version	BCD	SDCOM-SW-Modul Versioninformation
5	SDCOM Sub Version	BCD	SDCOM-SW-Modul Versioninformation
6	SDCOM Building Date – Year (MSB)	BCD	SDCOM-SW-Modul Versioninformation
7	SDCOM Building Date – Year (LSB)	BCD	SDCOM-SW-Modul Versioninformation
8	SDCOM Building Date – Month	BCD	SDCOM-SW-Modul Versioninformation
9	SDCOM Building Date – Day	BCD	SDCOM-SW-Modul Versioninformation
10	SDCOM Configuration Nr. (MSB)	HEX	Version Number of the Configuration-Database
11	SDCOM Configuration Nr. (LSB)	HEX	Version Number of the Configuration-Database
12	SDCOM Configuration Reference Nr. (MSB)	HEX	Version Reference Number of the Configuration-Database
13	SDCOM Configuration Reference Nr. (LSB)	HEX	Version Reference Number of the Configuration-Database
14	Checksum (Byte 0)	HEX	24-Bit Checksum for internal Versionhandling Bit 1-8
15	Checksum (Byte 1)	HEX	24-Bit Checksum for internal Versionhandling Bit 9-16
16	Checksum (Byte 2)	HEX	24-Bit Checksum for internal Versionhandling Bit 17-24
17	Reserved	HEX	
18	Reserved	HEX	

Table 4.4.4-16 System Diagnostic General Parameter Data

Note: The data corresponding to local identifier $E8 begins with byte 2 as byte 0 is reserved for the diagnostic service and byte 1 is reserved for the local identifier.

SYSTEM DIAGNOSTIC GLOBAL PARAMETER DATA ($E9)

Upon requesting $E9 with the diagnostic service **Read Data By Local Identifier ($21)**, the ECU shall return the information about the existent global process data. The number of bytes depends on the number of global process data.

Byte	Title	Encoding	Description
2	Number of Global Analog Values	HEX	
3	Number of Global States	HEX	

4	First Position in CAN-Data-Frame (MSB)	HEX	
5	First Position in CAN-Data-Frame (LSB)	HEX	
6	Data ID Global Process Data 1 (MSB)	HEX	
7	Data ID Global Process Data 1 (LSB)	HEX	
8	Timebase Global Process Data 1 (MSB)	HEX	
9	Timebase Global Process Data 1 (LSB)	HEX	
11	Reserved	HEX	
12	Reserved	HEX	
13	Size of Global Process Data 1	HEX	
14	Data ID Global Process Data 2 (MSB)	HEX	
15	Data ID Global Process Data 2 (LSB)	HEX	
16	Timebase Global Process Data 2 (MSB)	HEX	
17	Timebase Global Process Data 2 (LSB)	HEX	
18	Reserved	HEX	
19	Reserved	HEX	
20	Size of Global Process Data 2	HEX	
:		:	
n-6	Data ID Global Process Data n (MSB)	HEX	
n-5	Data ID Global Process Data n (LSB)	HEX	
n-4	Timebase Global Process Data n (MSB)	HEX	
n-3	Timebase Global Process Data n (LSB)	HEX	
n-2	Reserved	HEX	
n-1	Reserved	HEX	
n	Size of Global Process Data n	HEX	

Table 4.4.4-17 System Diagnostic Global Parameter Data

Note: The data corresponding to local identifier $E9 begins with byte 2 as byte 0 is reserved for the diagnostic service and byte 1 is reserved for the local identifier.

ECU CONFIGURATION ($EA)

Upon requesting $EA with the diagnostic service **Read Data By Local Identifier ($21)**, the ECU shall return the information about it's configuration. The format of the Record Data shall be binary.

DIAGNOSTIC PROTOCOL INFORMATION ($EB)

Upon requesting $EB with the diagnostic service **Read Data By Local Identifier ($21),** the ECU will return a positive response with the structure shown in Table 3.8.3-1.

Byte	Title	Encoding	Description
2	DCX Keyword Protocol 2000 Requirements Definition Version	HEX	Example: $22
3	DCX ECU Flash Reprogramming Requirements Definition Version	HEX	Example: $31
4	ECU Diagnostic Level Supported	HEX	Example: Level 3 = $03

Table 4.4.4-18 Diagnostic Protocol Information

4.4.5 DYNAMICALLY DEFINED LOCAL IDENTIFIER

This parameter identifies a new data record which has been defined by the diagnostic tool in the **Dynamically Define Local Identifier Request ($2C)** service. The new data record shall be read with the **Read Data By Local Identifier ($21)** service.

4.4.6 DEFINITION MODE

A new data record, or **Dynamically Defined Local Identifier**, can be built from other sources of data such as **Record Local Identifiers**. The source of the data is specified by the **Definition Mode**. In other words, the

Definition Mode defines if the data is to be taken from a **Record Local Identifier** when constructing a **Dynamically Defined Local Identifier**. In addition, the **Definition Mode** parameter is also used to indicate that a **Dynamically Defined Local Identifier** is to be cleared.

DEFINE BY LOCAL IDENTIFIER ($01)
The purpose of this **Definition Mode** is to reference data that is contained within one of the data records of a predefined local identifier.

DEFINE BY MEMORY ADDRESS ($02)
The purpose of this **Definition Mode** is to reference data that is contained within memory.

DEFINE BY IDENTIFIER ($03)
The purpose of this **Definition Mode** is to reference data that is contained within one of the data records of a predefined 2-byte identifier.

CLEAR DYNAMICALLY DEFINED LOCAL IDENTIFIER ($04)
The purpose of this **Definition Mode** is to clear a **Dynamically Defined Local Identifier** data record which has been previously defined in the ECU.

4.4.7 POSITION IN DYNAMICALLY DEFINED LOCAL IDENTIFIER
Identifies the data record position in the dynamically defined data record. The record specified may be a "1 byte" parameter (e.g. Engine Coolant Temperature) or a multiple byte record representing many parameters (e.g. analog and discrete inputs).

4.4.8 POSITION IN LOCAL IDENTIFIER
Identifies the data record position in the **Record Local Identifier** parameter which is stored in the ECU's memory.

4.4.9 POSITION IN IDENTIFIER
Identifies the data record position in the **Identifier** parameter which is stored in the ECU's memory.

4.4.10 RECORD VALUE
Used to return stored data to the diagnostic tool. Returnable data includes analog input and output signals, digital input and output signals, internal data and system status information if supported by the ECU.

4.5 READ MEMORY BY ADDRESS, WRITE MEMORY BY ADDRESS

4.5.1 MEMORY ADDRESS
Identifies the starting address in the ECU's memory where the read or write function is to be performed. This parameter may be considered a 3 byte identifier to define the location of stored data.

4.5.2 MEMORY SIZE
Specifies the number of bytes to read/write starting at a specified **Memory Address** in the ECU's memory.

4.6 SECURITY ACCESS

4.6.1 ACCESS MODE
Indicates to the ECU the step in progress of Security Access, the level of security the diagnostic tool wants to access, and the format of **Seed** and **Key** parameters. Two options are available for **Access Mode**: **Request Seed** and **Send Key**.

REQUEST SEED
Used by the diagnostic tool to request a **Seed** from an ECU. The value shall be an odd number greater than $01 if additional levels of security are supported. The default value is $01. (see Table 3.11.2-1)

SEND KEY
Used by the diagnostic tool to send a **Key** to the ECU. This parameter option is used after the diagnostic tool has requested a **Seed** and generated the **Key** to "unlock" the ECU. The **Send Key** data value shall be an even number one greater than the **Request Seed** parameter used in the **Security Access (Send Key) Request** message. (see Table 3.11.5-1).

4.6.2 KEY
Data sent from the diagnostic tool to "unlock" the ECU.

4.6.3 SEED
The Seed is sent by an ECU to the diagnostic tool so that the diagnostic tool may generate a Key to unlock the ECU. If the ECU returns all zero's to the diagnostic tool, then the ECU is not locked.

4.6.4 SECURITY ACCESS STATUS
Used to receive the status of the ECU security system.

4.7 DISABLE NORMAL MESSAGE TRANSMISSION, ENABLE NORMAL MESSAGE TRANSMISSION

4.8 RESPONSE REQUIRED
This parameter is used to specify whether or not it is necessary for an ECU to return a response.

4.8.1 NO RESPONSE REQUIRED
This parameter option indicates to the ECU that no response is required. In other words, no positive or negative response shall be sent when this parameter option is used.

4.8.2 RESPONSE REQUIRED
This parameter option indicates to the ECU that a response is required.

4.9 INPUT OUTPUT CONTROL BY LOCAL IDENTIFIER

4.9.1 LOCAL IDENTIFIER
Used by the **Input Output Control By Local Identifier Request ($30)** service to identify a local input signal, internal parameter, or output signal. These local identifiers are completely independent of the local identifiers specified in **Read Data by Local Identifier ($21)** service.

4.9.2 CONTROL STATE
Used to pass the desired state of the ECU's local input signals, internal parameters, and output signals.

4.9.3 INPUT OUTPUT CONTROL PARAMETER
Used by the **Input Output Control By Local Identifier Request ($30)** service to describe how the ECU has to control its inputs or outputs.

RETURN CONTROL TO ECU ($00)
This value shall indicate to the ECU that the diagnostic tool no longer has control over the input signal, internal parameter or output signal referenced by the **Input Output Local Identifier**.

REPORT CURRENT STATE ($01)
This value shall indicate to the ECU that it is requested to report the current state of the input signal, internal parameter or output signal referenced by the **Input Output Local Identifier**.

RESET TO DEFAULT ($04)
This value shall indicate to the ECU that it is requested to reset the input signal, internal parameter or output signal referenced by the **Input Output Local Identifier** to its default state.

FREEZE CURRENT STATE ($05)
This value shall indicate to the ECU that it is requested to freeze the current state of the input signal, internal parameter, or output signal referenced by the **Input Output Local Identifier**.

SHORT TERM ADJUSTMENT ($07)
This value shall indicate to the ECU that it is requested to adjust the input signal, internal parameter or output signal referenced by the **Input Output Local Identifier** in RAM to the value(s) included in the **Control Option** parameter(s). With return to Default Session or Return Control to ECU the ECU resumes control over this value. (E.g. set Idle Air Control Valve to a specific step number, set pulse width of valve to a specific value/duty cycle).

LONG TERM ADJUSTMENT ($08) (DCS ONLY)
This value shall indicate to the ECU that it is requested to adjust the input signal, internal parameter or output signal referenced by the **Input Output Local Identifier** in EEPROM/FLASH EEPROM to the value(s) included in the **Control Option** parameter(s). (E.g. set Engine Idle Speed, set CO Adjustment parameter to a specific value).

4.10 START/STOP/REQUEST ROUTINE RESULTS BY LOCAL IDENTIFIER

4.10.1 ROUTINE ENTRY OPTION
Used to specify the start conditions of a routine.

4.10.2 ROUTINE ENTRY STATUS
Used by the **Start Routine By Local Identifier Positive Response ($71)** service to give the diagnostic tool additional information about the status of the ECU following the start of the routine.

4.10.3 ROUTINE EXIT OPTION
Used to specify the exit conditions of the routine.

4.10.4 ROUTINE EXIT STATUS
Used by the **Stop Routine By Local Identifier Positive Response ($72)** service to give the diagnostic tool additional information about the status of the ECU following the exit of the routine. The available options are as follows:

NORMAL EXIT WITH RESULTS AVAILABLE ($61)

NORMAL EXIT WITHOUT RESULTS AVAILABLE ($62)

ABNORMAL EXIT WITHOUT RESULTS AVAILABLE ($64)

4.10.5 ROUTINE LOCAL IDENTIFIER
Used to define the ECU's local routine.

FLASH ERASE ROUTINE ($E0)
Upon requesting $E0 with the diagnostic service **Start Routine By Local Identifier ($31)**, the ECU shall start the Flash Erase Routine. For more information regarding the Flash Erase Routine, please refer to Reference "*L"* in "Appendix A – *References*".

FLASH CHECK ROUTINE ($E1)
Upon requesting $E1 with the diagnostic service **Start Routine By Local Identifier ($31)**, the ECU shall start the Flash Check Routine. For more information regarding the Flash Check Routine, please refer to Reference "*L"* in "Appendix A – *References*".

TELL-TALE RETENTION STACK ($E2)
Upon requesting $E2 with the diagnostic service **Return Routine Results By Local Identifier ($33)**, the ECU shall return the tell-tale retention stack. The structure of the positive response message shall follow Table 3.19.3-1 and the content of the local identifier will match Table 4.10.5-1. The diagnostic trouble codes shall be reported in the J2012 format. For more information regarding Tell-tale, please refer to document "H " in "Appendix A – *References*".

Byte	Title	Encoding	Description
2	DTC Read Counter	DEC	Increment each time DTC's read with SID $18
3	Odometer at time of last DTC Read (MSB)	DEC	Odometer value at time DTC's were last read with SID $18 (MSB)
4	Odometer at time of last DTC Read (LSB)	DEC	Odometer value at time DTC's were last read with SID $18 (LSB)
5	DTC #1 (MSB)	J2012 HEX	
6	DTC #1 (LSB)	J2012 HEX	
7	Odometer at time of DTC#1 stored (MSB)	DEC	
8	Odometer at time of DTC#1 stored (LSB)	DEC	
9	DTC #2 (MSB)	J2012 HEX	
10	DTC #2 (LSB)	J2012 HEX	
11	Odometer at time of DTC#2 stored (MSB)	DEC	
12	Odometer at time of DTC#2 stored (LSB)	DEC	
	:		
32	DTC #8 (MSB)	J2012 HEX	
33	DTC #8 (LSB)	J2012 HEX	
34	Odometer at time of DTC#8 stored (MSB)	DEC	
35	Odometer at time of DTC#8 stored (LSB)	DEC	

Table 4.10.5-1 Tell-tale Information

Note: The data corresponding to local identifier $E2 begins with byte 2 as byte 0 is reserved for the diagnostic service and byte 1 is reserved for the local identifier.

REQUEST DTCS FROM SHADOW ERROR MEMORY ($E3)

This value is used by the diagnostic tool to indicate to the ECU that the ECU shall respond with the *diagnostic trouble codes (DTCs) in 2 byte format and their status* that are stored in an additional error memory that can not be erased by Clear Diagnostic Information (SID $14). This error memory mirrors the normal error memory and can be used for example if the normal error memory is erased. The structure of the positive response message shall follow Table 3.17.3-1 and the content of the local identifier will match Table 4.10.5-2.
Note: For the shadow error memory the service $31 $E3 $FF $00 is the equivalent to the service $18 $02 $FF $00 for the normal error memory.

Byte	Title	Encoding	Description
2	**Number OF DTC's**	HEX	
3	**Diagnostic Trouble Code #1 {High Byte }**	HEX	
4	**Diagnostic Trouble Code #1 {Low Byte }**	HEX	
5	**Status Of DTC#1**	HEX	
:	**:**		
N-2	**Diagnostic Trouble Code #m {High Byte }**	HEX	
N-1	**Diagnostic Trouble Code #m {Low Byte }**	HEX	
N	**Status Of DTC #m**	HEX	

Table 4.10.5-2 DTCs from Shadow Error Memory

Note: The data corresponding to local identifier $E3 begins with byte 2 as byte 0 is reserved for the diagnostic service and byte 1 is reserved for the local identifier.

REQUEST ENVIRONMENT DATA FROM SHADOW ERROR MEMORY ($E4)

This value is used by the diagnostic tool to indicate to the ECU that the ECU shall respond with the *environment data corresponding to the indicated DTC* that is stored in an additional error memory. This error memory can not be erased by Clear Diagnostic Information (SID $14). This error memory mirrors the normal error memory and can be used for example if the normal error memory is erased. The structure of the positive response message shall follow Table 3.17.3-1 and the content of the local identifier will match Table 4.10.5-3.
Note: For the shadow error memory the service $31 $E4 {DTC-High Byte} {DTC-Low Byte} is the equivalent to the service $17 {DTC-High Byte} {DTC-Low Byte}

Byte	Title	Encoding	Description
2	**Number OF DTC**	HEX	$01 if Environment Data for the requested DTC is available $00 if no Environment Data for the requested DTC is available
3	**Diagnostic Trouble Code { High Byte }**	HEX	The **Diagnostic Trouble Code** parameters must be identical to the DTC sent in the request message.
4	**Diagnostic Trouble Code { Low Byte }**	HEX	
5	**Status Of DTC**	HEX	
6	**Environment Data #1**	HEX	
:	**:**	:	
N	**Environment Data #m**	HEX	

Table 4.10.5-3 Environment Data from Shadow Error Memory

Note: The data corresponding to local identifier $E4 begins with byte 2 as byte 0 is reserved for the diagnostic service and byte 1 is reserved for the local identifier.

REQUEST EVENT INFORMTAION ($E5)

This Routine Identifier shall be used to request the event identification numbers. The format of request and response shall follow Request DTCs From Shadow Error Memory ($31 $E3) but with the Routine Local Identifier $E5.

REQUEST EVENT ENVIRONMENT DATA ($E6)

This Routine Identifier shall be used to request additional information stored with event identification numbers. The format of request and response shall follow Request Environment Data From Shadow Error Memory ($31 $E4) but with the Routine Local Identifier $E6.

REQUEST SOFTWARE MODULE INFORMATION ($E7)

This Routine Identifier shall be used to request information of the standard software modules. This information shows details about the software modules in use and is for development purposes.

CLEAR TELL-TALE RETENTION STACK ($E8)

This Routine Identifier is used to clear the Tell-Tale Retention Stack. Upon requesting $E8 with the diagnostic service **Start Routine By Local Identifier ($31),** the ECU will return a positive response with the structure shown in Table 3.8.3-1

Byte	Title	Encoding	Description
2	**Status of Clearing Tell-Tale Retention Stack**	HEX	$00 = Failed, $01=Passed

Table 4.10.5-4 Clear Tell-Tale Retention Stack

CLEAR EVENT INFORMATION ($E9)

This Routine Identifier is used to clear the Event Information memory. After successfully erasing the event information memory, the ECU will return a positive response.

4.10.6 ROUTINE RESULTS

Used by the **Request Routine Results By Local Identifier Positive Response ($73)** service to provide results (exit status info) of the routine which has been stopped previously in the ECU.

4.11 REQUEST DOWNLOAD, REQUEST UPLOAD, TRANSFER DATA, REQUEST TRANSFER EXIT

4.11.1 DATA FORMAT IDENTIFIER

A 1 byte identifier used to specify the compression method and the encryption method. The high nibble specifies the Compression Method and the low nibble specifies the encrytion method. Use of encoding improves tamper protection.

COMPRESSED DATA

If the first nibble of the Data Format Identifier is a value other than $0, then the data is compressed regardless of the second nibble. Define compressed

ENCRYPTED DATA
If the second nibble of the Data Format Identifier is a value other than $0, then the data is encrypted regardless of the first nibble. Define encrypted

UNCOMPRESSED DATA
If the first nibble of the Data Format Identifier is a $0, then the data is uncompressed regardless of the second nibble.

UNENCRYPTED DATA
If the second nibble of the Data Format Identifier is a $0, then the data is unencrypted regardless of the first nibble.

4.11.2 MAX NUMBER OF BLOCK LENGTH
When used by the **Request Upload Positive Response ($75)** or the **Request Download Positive Response ($74)**, the ECU informs the diagnostic tool of how many data bytes shall be included in each **Transfer Data Request ($36)** service. This parameter allows the diagnostic tool to adapt to the receive buffer size of the ECU before it starts transferring data to the ECU. The Max Number of Block Length shall include the service identifier byte and the optional **Block Sequence Counter**. It shall not exceed the maximum number of bytes that can be transferred on the data link that is use (e.g. for CAN: 4095 bytes).

4.11.3 BLOCK SEQUENCE COUNTER
The **Block Sequence Counter** parameter value starts at $01 with the first **Transfer Data** request that follows the **Request Download ($34)** or **Request Upload ($35)** service. Its value is incremented by 1 for each subsequent **Transfer Data** request. At the value of $FF, the **Block Sequence Counter** rolls over and starts at $00 with the next **Transfer Data** request message.
Example use cases:
a) If a **Transfer Data** request to download data is correctly received and processed in the ECU but the positive response message does not reach the tester, then the tester would determine an application layer timeout and would repeat the same request (including the same **Block Sequence Counter**). The ECU would receive the repeated **Transfer Data** request and could determine based on the included **Block Sequence Counter** that this **Transfer Data** request is repeated. The ECU would send the positive response message immediately without writing the data once again into its memory.
b) If the **Transfer Data** request to download data is not received in the ECU then the ECU would not send a positive response message. The tester would determine an application layer timeout and would repeat the same request (including the same **Block Sequence Counter**). The ECU would receive the repeated **Transfer Data** request and could determine based on the included **Block Sequence Counter** that this is a new **Transfer Data**. The ECU would process the service and would send the positive response message.
c) If a **Transfer Data** request to upload data is correctly received and processed in the ECU but the positive response message does not reach the tester then the tester would determine an application layer timeout and would repeat the same request (including the same **Block Sequence Counter**). The ECU would receive the repeated **Transfer Data** request and could determine based on the included **Block Sequence Counter** that this **Transfer Data** request is repeated. The ECU would send the positive response message immediately accessing the previously provided data once again in its memory.
d) If the **Transfer Data** request to upload data is not received in the ECU then the ECU would not send a positive response message. The tester would determine an application layer timeout and would repeat the same request (including the same **Block Sequence Counter**). The ECU would receive the repeated **Transfer Data** request and could determine based on the included **Block Sequence Counter** that this is a new **Transfer Data**. The ECU would process the service and would send the positive response message.

4.11.4 TRANSFER RESPONSE PARAMETER
Used by the **Transfer Data Positive Response ($76)** service to pass data if **Request Upload ($34)** service precedes the **Transfer Data ($36)** service. If **Request Download ($34)** does not precede the **Transfer Data ($36**), then this parameter does not need to be included in the **Transfer Data Positive Response ($76)** message**.**

4.11.5 TRANSFER REQUEST PARAMETER
Used by the **Transfer Data Request ($36)** service to pass data if **Request Download ($34)** service precedes the **Transfer Data ($36)** service. If **Request Upload ($35)** service does not precede the **Transfer Data ($36**) service, then this parameter does not need to be included in the **Transfer Data Request ($36)** service**.**

4.11.6 UNCOMPRESSED MEMORY SIZE

A 3 byte value to indicate the uncompressed data size used by the ECU to allocate a sufficient amount of memory.

4.12 RESPONSE ON EVENT

4.12.1 EVENT WINDOW TIME PARAMETER DEFINITION

TESTER PRESENT REQUIRED ($01)

The Event Window Time is set to **P3**. (P3 is defined as the maximum time between the 2 consecutive Tester Present messages. This value is data link specific.) The requested functionality shall remain active as long as the requesting test tool sends Tester Present messages within the Event Window Time. Each received Tester Present message shall restart the Event Window Time.

INFINITE TIME TO RESPONSE ($02)

This value specifies that the event window shall stay active for an infinite amount of time (i.e. open window until power off, reset, or request Stop Response On Event).

NO EVENT WINDOW ($80)

This value specifies that no event window is necessary because for the specified Event Type no Service to Respond is defined and no event will occur (e.g. Event Type Stop Response On Event or Report Activated Events).

TIME INTERVAL ($03 - $7F)

The Event Window Time is set to the value of this parameter. (I.e. The Event Window remains open for 3 to 127 time units.) The resolution is TBD.

4.12.2 EVENT TYPE DEFINITION

REPORT ACTIVATED EVENTS ($80)

This value is used to indicate that in the positive response all events are reported that have been activated in the ECU with the Response on Event Service (and are currently active).
This Event Type requires 0 additional Event Type Parameters.

STOP RESPONSE ON EVENT ($81)

This value is used to stop the server sending responses on event.
This Event Type requires 0 additional Event Type Parameters.

ON NEW DTC ($82)

This value identifies the event as a new DTC is detected.
This Event Type requires 1 additional Event Type Parameter.

Parameter Value	Description
$00	Reserved
$01	Pending
$02	Active
$03	Stored
$04-$FF	ECU Specific

Table 4.12.2-1 On New DTC

ON TIMER INTERRUPT ($83)

This value identifies the event as a timer interrupt. This Event Type may be used to send messages periodically.
This Event Type requires 2 additional Event Type Parameters that describe the interrupt identifier and time.

Parameter Value	Description
$0000	Reserved
$0001 - $FFFF	ECU Specific

Table 4.12.2-2 On Timer Interrupt

ON CHANGE OF RECORD VALUE ($84)

This value identifies the event as changes in internal record values identified by Record Data Identifier.
This Event Type requires 1 additional Event Type Parameter. This parameter is an identifier to specify the type of change.
The following table is an example of how the Event Type Parameter may work:

Parameter Value	Description
$00	Reserved
$01	Bit 5, Byte 3, for the Record Value of local identifier $0A, changes from 0 to 1
$02	Byte 5 for the Record Value of local identifier $F3, threshold $F0 is passed.
$03	Bytes 3 and 4, for the Record Value of local identifier $11, equals $FFFF

Table 4.12.2-3 On Change of Record Value

ON COMPARISON OF VALUES ($A0)

A specific change of a data value defines a data value event. The event shall be reported once when it occurs. A current data vale (CDV) is compared to a parameter (P). Every supported LID of the ECU can be used for the CDV.
This Event Type requires 9 additional Event Type Parameter.

Parameter Value	Description
$XX	Event Type Parameter 1 = Local Identifier
$XX	Event Type Parameter 2 = Valueinfo #1
$XX	Event Type Parameter 3 = Valueinfo #2
$XX	Event Type Parameter 4 = Operator
$XX	Event Type Parameter 5 = Comparison Parameter Byte 4
$XX	Event Type Parameter 6 = Comparison Parameter Byte 3
$XX	Event Type Parameter 7 = Comparison Parameter Byte 2
$XX	Event Type Parameter 8 = Comparison Parameter Byte 1
$XX	Event Type Parameter 9 = Hysteresis [%]

Table 4.12.2-3 On Comparison of Values

4.13 DTC SETTING MODE

This parameter is used to control the setting of DTCs.

4.13.1 ON ($01)

The ECU(s) shall enable the setting of diagnostic trouble codes according to normal operating conditions.

4.13.2 OFF ($02)

The ECU(s) shall disable the setting of diagnostic trouble codes.

4.14 DOCUMENT RELATED DEFINITIONS

4.14.1 RESERVED FOR FUTURE DEFINITION BY DCX

This range of values is reserved for Vehicle Diagnostics, CoC and Diagnostic Standards (EP / EID). Future revisions of this document may include additional parameters. No values within the specified range may be defined for use with production intent without consent of Vehicle Diagnostics, CoC and Diagnostic Standards (EP / EID).

4.14.2 RESERVED FOR FUTURE DEFINITION BY ISO

This range of values is reserved by ISO. Future revisions of this document may include additional parameters. No values within the specified range may be defined for use with production intent.

4.14.3 SYSTEM SUPPLIER SPECIFIC

This range of values may be defined by the supplier for supplier specific use.

5 NEGATIVE RESPONSE CODES

The following table lists and assigns hex values for all response codes used in KWP 2000.

Note: The ECU shall use negative response messages if the ECU cannot respond with a positive response message on a diagnostic tool request message. In such a case, the ECU shall send one of the response codes listed below as specified in Figure 5-1.

The figure below specifies the ECU behavior on a diagnostic tool request message. This figure shows the logic as specified in the description of the response codes and to be implemented in the ECU and diagnostic tool as appropriate.

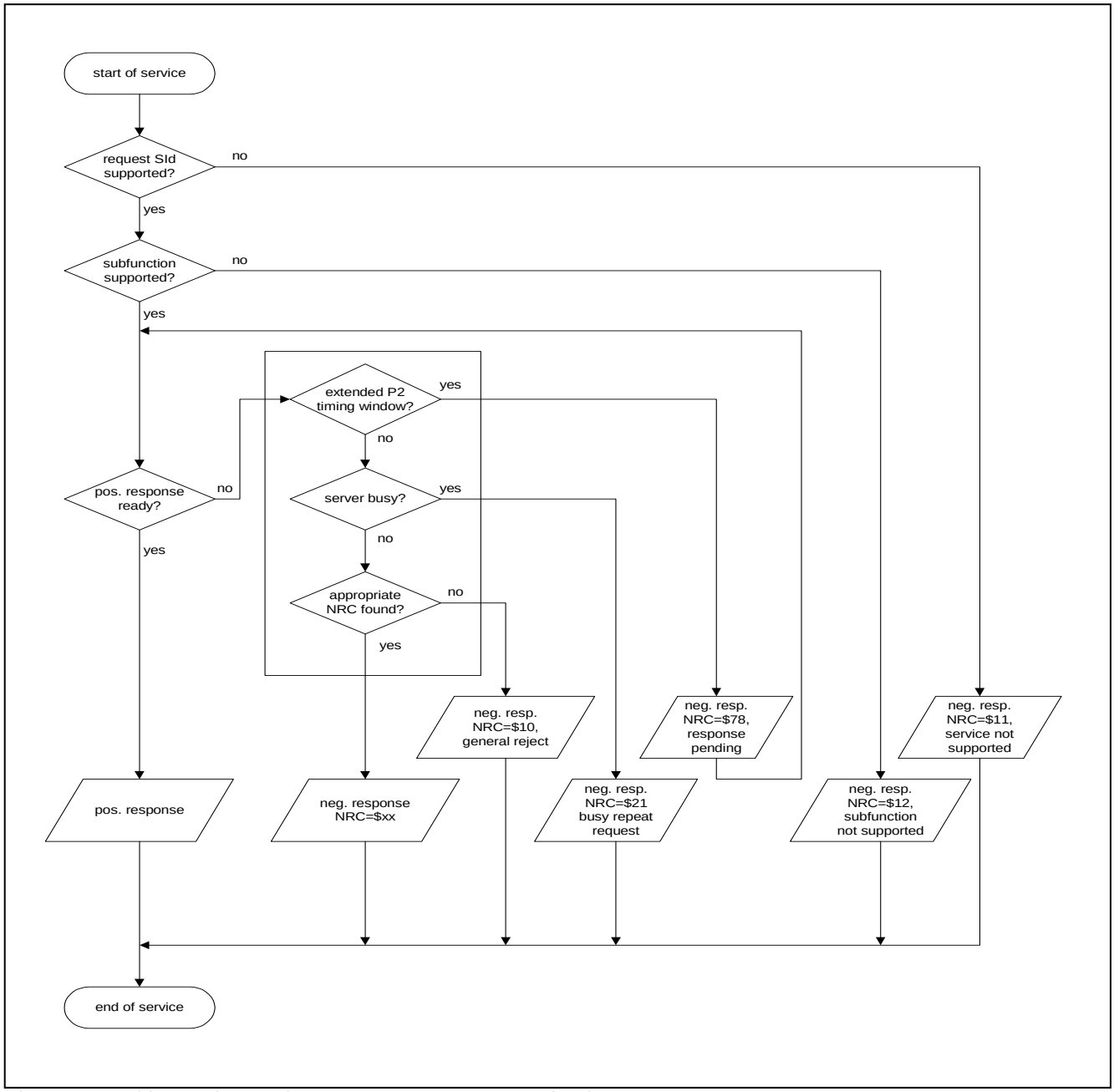

Figure 7 Positive and Negative Response Message Behavior

5.1 GLOBAL NEGATIVE RESPONSE CODES

The following negative response codes may be used globally except as noted:

- General Reject ($10)
- Service Not Supported ($11)
- Sub Function Not Supported / Invalid Format ($12)
- Busy / Repeat Request ($21)
- Conditions Not Correct Or Request Sequence Error ($22) [Excludes: **Tester Present ($3E)**]
- Security Access Denied / Security Access Requested ($33) [Excludes: **Start Diagnostic Session ($10)** and **Tester Present ($3E)**]
- Request Correctly Received / Response Pending ($78)
- Service Not Supported In Active Diagnostic Session ($80)) [Excludes: **Start Diagnostic Session ($10)** and **Tester Present ($3E)**

5.2 NEGATIVE RESPONSE CODE $10 – GENERAL REJECT

DESCRIPTION:
This response is used only when no other negative response fits. It is **NOT** to be implemented as the only available negative response code when a service ID is requested. It's primary function is to serve as an escape code when no other negative response is valid.
The communication timing is not affected by this response code.

APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S):
All

EXAMPLE:
The ECU shall send this response code if no other response code is available which properly indicates the rejection. It is up to the scan tool manufacturer to define the reaction of the diagnostic tool in case this response code is sent by the ECU. If a repetition of the request message is performed the number of repetitions shall be limited by the scan tool manufacturer to a certain value to prevent deadlock states.

5.3 NEGATIVE RESPONSE CODE $11 – SERVICE NOT SUPPORTED

DESCRIPTION:
This response code indicates that the requested action will not be taken because the ECU does not support the requested diagnostic service.
The communication timing is not affected by this response code.

APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S):
All

EXAMPLE:
The ECU shall send this response code in case the diagnostic tool has sent a request message with a service identifier which is either unknown (not a KWP 2000 Service Identifier) or not supported by the ECU.

5.4 NEGATIVE RESPONSE CODE $12 – SUB FUNCTION NOT SUPPORTED / INVALID FORMAT

DESCRIPTION:
This response code indicates that the requested action will not be taken because the ECU does not support the arguments of the request message or the format of the argument bytes do not match the prescribed format for the specified service.
The communication timing is not affected by this response code.

APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S):
All

EXAMPLE:
The ECU shall send this response code in case the diagnostic tool has sent a request message with a known and supported service identifier but with "sub parameters" which are either unknown or not supported or have an invalid format. It is recommended that the diagnostic tool shall not repeat the identical request message.

5.5 NEGATIVE RESPONSE CODE $21 – BUSY / REPEAT REQUEST

DESCRIPTION:
This response code indicates that the ECU is temporarily too busy to perform the requested operation. In this circumstance repetition of the "identical request message" or "another request message" shall be performed by the diagnostic tool. This response code shall be returned for example while an ECU is in the process of clearing stored DTC(s) information or fetching information.
The communication timing is not affected by this response code.

APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S):
All

EXAMPLE:
The ECU shall send this response code in case the diagnostic tool has sent a request message at a time when the ECU is busy with internal processing not related to the current request message. It is recommended that the diagnostic tool shall repeat the request message within the **P3** timing window.

5.6 NEGATIVE RESPONSE CODE $22 – CONDITIONS NOT CORRECT OR REQUEST SEQUENCE ERROR

DESCRIPTION:
This response code indicates that the requested action will not be taken because the ECU prerequisite conditions are not met. This request may occur when sequence sensitive requests are issued in the wrong order.
The communication timing is not affected by this response code.

APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S):
All (Excluding **Tester Present($3E)**)

EXAMPLE:
The ECU shall send this response code in case the diagnostic tool has sent a known and supported request message at a time when the ECU has expected another request message because of a predefined sequence of services. A typical example of occurrence is the **Security Access** service which requires a sequence of messages as specified in the message description of this service.

5.7 NEGATIVE RESPONSE CODE $23 – ROUTINE NOT COMPLETE

This negative response code shall no longer be used

5.8 NEGATIVE RESPONSE CODE $31 – REQUEST OUT OF RANGE

DESCRIPTION:
This response code indicates that the requested action will not be taken because the ECU detects the request message contains a data byte(s) which attempt(s) to substitute (a) value(s) beyond its range of authority (e.g. attempting to substitute a data byte of 111 when the data is only defined to 100).

The communication timing is not affected by this response code.

APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S):
$14, $17, $18, $23, $27, $2C, $30, $31, $32, $34, $35, $36, $3B, $3D

EXAMPLE:
The ECU shall send this response code in case the diagnostic tool has sent a request message including data bytes to adjust a variant which does not exist (invalid) in the ECU. This response code shall be implemented for all services which allow the diagnostic tool to write data or adjust functions by data in the ECU.

5.9 NEGATIVE RESPONSE CODE $33 – SECURITY ACCESS DENIED / SECURITY ACCESS REQUESTED

DESCRIPTION:
This response code indicates that the requested action will not be taken because the ECU's security strategy has not been satisfied by the diagnostic tool.

The communication timing is not affected by this response code.

APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S):
All (Excluding **Start Diagnostic Session($10)** and **Tester Present ($3E)**)

EXAMPLE:
The ECU shall send this response code if one of the following cases occur:
- The test conditions of the ECU are not met.
- The required message sequence (e.g. **Start Diagnostic Session**, **Security Access**) is not met
- The diagnostic tool has sent a request message which requires an unlocked ECU.

5.10 NEGATIVE RESPONSE CODE $35 – INVALID KEY

DESCRIPTION:
This response code indicates that security access has not been given by the ECU because the key sent by the diagnostic tool did not match with the key in the ECU's memory. This counts as an attempt to gain security. The ECU shall remain locked.
The communication timing is not affected by this response code.

APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S):
Security Access ($27) (Send Key)

EXAMPLE:
The ECU shall send this response code in case the diagnostic tool has sent a **Security Access** request message with the **Send Key** and **Key** parameter where the key value does not match the key value stored in the ECU's memory. The ECU shall increment it's internal **Security Access Failed** counter.

5.11 NEGATIVE RESPONSE CODE $36 – EXCEED NUMBER OF ATTEMPTS

DESCRIPTION:
This response code indicates that the requested action will not be taken because the diagnostic tool has unsuccessfully attempted to gain security access more times than the ECU's security strategy will allow. Refer to message description of the **Security Access** service definition.

The communication timing is not affected by this response code.

APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S):
Security Access ($27) (Request Seed)

EXAMPLE:
The ECU shall send this Negative Response Code in case the diagnostic tool has sent a **Security Access** request message with the **Send Key** and **Key** parameter where the key value does not match the key value stored in the ECU's memory and the number of attempts (**Security Access Failed** counter value) have reached the ECU's **Security Access Failed** calibration value.

5.12 NEGATIVE RESPONSE CODE $37 –REQUIRED TIME DELAY NOT EXPIRED

DESCRIPTION:
This response code indicates that the requested action will not be taken because the diagnostic tool's latest attempt to gain security access was initiated before the ECU's required timeout period had elapsed.

The communication timing is not affected by this response code.

APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S):
Security Access ($27) (Request Seed)

EXAMPLE:
An invalid Key requires the diagnostic tool to start over from the beginning with a **Security Access** service. If the security protection algorithm is not passed after "X" failed attempts ("X" = specified in the ECU's memory), all additional attempts are rejected for at least "Y" seconds ("Y" = specified in the ECU's memory). The "Y"

second timer shall begin with the "X" failed attempt or upon a power/ignition cycle or reset of the ECU to ensure a security lockout after all power interruptions.

5.13 NEGATIVE RESPONSE CODE $40 – DOWNLOAD NOT ACCEPTED

DESCRIPTION:
This response code indicates that an attempt to download to an ECU's memory cannot be accomplished due to some fault conditions.

The communication timing is not affected by this response code.

APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S):
Request Download ($34), Request Transfer Exit ($37)

EXAMPLE:
The ECU shall send this response code in case the diagnostic tool has sent a **Request Download** request message which the ECU can not accept and the failure can not be described with another negative response code.

5.14 NEGATIVE RESPONSE CODE $50 – UPLOAD NOT ACCEPTED

DESCRIPTION:
This response code indicates that an attempt to upload from an ECU's memory cannot be accomplished due to some fault conditions.
The communication timing is not affected by this response code.

APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S):
Request Upload ($35), Request Transfer Exit ($37)

EXAMPLE:
The ECU shall send this response code in case the diagnostic tool has sent a **Request Upload** request message which the ECU can not accept and the failure can not be described with another negative response code.

5.15 NEGATIVE RESPONSE CODE $71 – TRANSFER SUSPENDED

DESCRIPTION:
This response code indicates that a data transfer operation was halted due to some fault.

The communication timing is not affected by this response code.

APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S):
Transfer Data ($36), Request Transfer Exit ($37)

EXAMPLE:
The ECU shall send this response code in case the diagnostic tool has sent a request message with incorrect data. In such case this response code shall be sent if no other response code matches the situation.

5.16 NEGATIVE RESPONSE CODE $78 – REQUEST CORRECTLY RECEIVED / RESPONSE PENDING

DESCRIPTION:
This response code indicates that the request message was received correctly and that any parameters in the request message were valid, but the action to be performed may not be completed yet. This response code can be used to indicate that the request message was properly received and does not need to be re-transmitted, but the ECU is not yet ready to receive another request.
This response code shall only be used in a negative response message if the ECU will not be able to receive further request messages from the diagnostic tool within the **P3** timing window. This may be the case if the ECU does data processing or executes a routine which does not allow any attention to serial communication.
The following description specifies the communication timing method: This response code shall manipulate the **P2**max timing parameter value in the ECU and the diagnostic tool. The **P2**max timing parameter is set to the

value [in ms] of the **P3**max timing parameter. In addition, the diagnostic tool shall disable a point to point **Tester Present** service. As soon as the ECU has completed the task (routine) initiated by the request message it shall send either a positive or negative response message (negative response message with a response code other than $78) based on the last request message received. When the diagnostic tool has received the positive response message which has been preceded by the negative response message(s) with this response code, the diagnostic tool and the ECU shall reset the **P2** timing parameter to the previous **P2** timing value. In addition, the diagnostic tool shall re-enable the point to point **Tester Present** service. The diagnostic tool shall not repeat the request message after the reception of a negative response message with this response code. For additional timing information, please refer to the Network Transport Protocol Document. (See reference "J" in "Appendix A – *References*")

The communication timing is affected if this response code is used.

APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S):
All

EXAMPLE:
A typical example where this response code may be used is when the diagnostic tool has sent a request message which includes data to be programmed or erased in flash memory of the ECU. If the programming/erasing routine (usually executed out of RAM) routine is not able to support serial communication while writing to the flash memory the ECU shall send a negative response message with this response code. As soon as the data are programmed or erased in flash memory the ECU shall send a positive response message (or negative response message but not $78).

5.17 NEGATIVE RESPONSE CODE $80 – SERVICE NOT SUPPORTED IN ACTIVE DIAGNOSTIC SESSION

DESCRIPTION:
This response code indicates that the requested action will not be taken because the ECU does not support the requested service in the diagnostic session (session) currently active.
The communication timing is not affected by this response code.

APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S):
All (Excluding **Start Diagnostic Session($10)**, **Tester Present($3E)**)

EXAMPLE:
The ECU shall send this response code in case the diagnostic tool has sent a request message (**Request Download**) which is only supported in the programming session but the ECU is in a diagnostic session which does not support this service.

5.18 NEGATIVE RESPONSE CODES $81 TO $8F - RESERVED FOR FUTURE USE BY ISO 14230

DESCRIPTION:
This range of values is reserved by ISO 14230. Future revisions of this document may include additional negative response codes be added. No values within the specified range may be defined for use with production intent.

5.19 NEGATIVE RESPONSE CODES $90 TO $99 – TO BE DEFINED BY DCX DIAGNOSTICS

DESCRIPTION:
This range of values is reserved for DCX Diagnostics. Future revisions of this document may include additional response be added. No values within the specified range may be defined for use with production intent without consent of DCX Diagnostics.

5.20 NEGATIVE RESPONSE CODE $9A – DATA DECOMPRESSION FAILED

DESCRIPTION:
This response code indicates that the tester detected an error when decompressing the transferred data.

The communication timing is not affected by this response code.

APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S):
Transfer Data ($36), Request Transfer Exit ($37)

EXAMPLE:
If the ECU detects that the decompressed number of bytes is not equal to the expected number of bytes, it will send a negative response data decompression failed.

5.21 NEGATIVE RESPONSE CODE $9B – DATA DECRYPTION FAILED

DESCRIPTION:
This response code indicates that the tester detected an error when decrypting the transferred data.

The communication timing is not affected by this response code.

APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S):
Transfer Data ($36), Request Transfer Exit ($37)

EXAMPLE:
The encryption algorithm might require a defined data padding. If the download does not comply with the expected number of fill bytes the ECU will send a negative response data decryption failed.

5.22 NEGATIVE RESPONSE CODES $9C TO $99 – TO BE DEFINED BY DCX DIAGNOSTICS

DESCRIPTION:
This range of values is reserved for DCX Diagnostics. Future revisions of this document may include additional negative response codes be added. No values within the specified range may be defined for use with production intent without consent of DCX Diagnostics.

5.23 NEGATIVE RESPONSE CODE $A0 – ECU NOT RESPONDING

DESCRIPTION:
This response code indicates that the requested network-ECU does not respond within the specified time and therefore the gateway sends this response code.

The communication timing is not affected by this response code.

APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S):
All

EXAMPLE:
An ECU (ECU A), which acts as a gateway to another ECU (ECU B), receives a diagnostic request from a diagnostic tool and passes the request to ECU B on a bus (or sub-node) different than the one the request was received on. If ECU B does not respond, ECU A shall return $7F $XX $A0.

5.24 NEGATIVE RESPONSE CODE $A1 – ECU-ADDRESS UNKNOWN

DESCRIPTION:
This response code indicates that the addressed network – ECU is unknwon to the gateway.
The communication timing is not affected by this response code.

APPLICABLE TO THE FOLLOWING DIAGNOSTIC SERVICE(S):
All

EXAMPLE:
An ECU (ECU A), which acts as a gateway to another ECU (ECU B), receives a diagnostic request from a diagnostic tool but is unable to send the request because the address is unknown so the request can not be routet.

5.25 NEGATIVE RESPONSE CODES $A2 TO $F9 – TO BE DEFINED BY DCX DIAGNOSTICS

DESCRIPTION:
This range of values is reserved for DCX Diagnostics. Future revisions of this document may include additional negative response codes be added. No values within the specified range may be defined for use with production intent without consent of DCX Diagnostics

5.26 NEGATIVE RESPONSE CODE $FF - RESERVED FOR FUTURE USE BY ISO 14230

DESCRIPTION:
This range of values is reserved by ISO 14230. Future revisions of this document may include additional negative response codes be added. No values within the specified range may be defined for use with production intent.

APPENDIX A – *REFERENCES*

A. *ISO 14229, Road Vehicles – Diagnostic Systems – Diagnostic Services Specification*
B. *ISO 14230-3*, *Road Vehicles - Diagnostic Systems - Keyword Protocol 2000 - Part 3: Implementation of Diagnostic Services*
C. *Keyword Protocol 2000 - Part 3: Implementation of Diagnostic Services - Recommended Practice*, Keyword Protocol 2000 Task Force
D. Keyword Protocol 2000 v1.1, *Serielle Diagnose-Schnittstelle für elektronische Steuergeräte,* authored by Werner Preuschoff
E. *SAE J1979 / ISO 15031- 5 – E/E Diagnostic Test Modes*
F. *SAE J2012 / ISO 15031-6 – Diagnostic Trouble Codes Definitions*
G. *SAE J2186 – E/E Data Link Security*
H. *Diagnostic Performance Standard* - authored by Vehicle Diagnostics, CoC
I. *Implementation Regulations Diagnosis (AV-Diagnose)* – authored by Diagnostic Standards, DCS
J. *Network Transport Protocol (ISO 15765-2) Requirements Definition,* authored by Vehicle Diagnostics, CoC
K. *Diagnostic Trouble Code Requirements Definition*, authored by Vehicle Diagnostics, CoC
L. *ECU Flash Reprogramming Requirements Specification* - authored by Vehicle Diagnostics, CoC and Diagnostic Standards (GSP/SDE), Version 3.1
M. *Flash Loader Specification (Software Module)* – authored by EP/EKP and Core Body Electronics
N. *Supplier Identification Reference* - authored by Vehicle Diagnostics, CoC and Diagnostic Standards (GSP/SDE)
O. *VIN Coding Guide* – Authored by Michael Fischer – DaimlerChrysler Corporation
P. *DCA Security Application Guideline* – Authored by Vehicle Diagnostics, CoC
Q. *DCX MMC Common State Encoding Tables* – Authored by Vehicle Diagnostics, CoC and Diagnostic Standards (GSP/SDE)